TUGAS AKHIR – TE 141599

STRATEGI PENYERANGAN NPC BERBASIS *RULE* PADA *GAME REAL TIME STRATEGY*

I Putu Widya Wiranata
NRP 2212106028

Dosen Pembimbing
Mochamad Hariadi, ST., M.Sc., Ph.D.
Dr. Supeno Mardi Susiki Nugroho, ST., MT.

JURUSAN TEKNIK ELEKTRO
Fakultas Teknologi Industri
Institut Teknologi Sepuluh November
Surabaya 2015

FINAL PROJECT – TE 141599

ATTACK STRATEGY OF NPC RULE BASED ON REAL TIME STRATEGY GAME

I Putu Widya Wiranata
NRP 2212106028

Advisor
Mochamad Hariadi, ST., M.Sc., Ph.D.
Dr. Supeno Mardi Susiki Nugroho, ST., MT.

DEPARTMENT OF ELECTRICAL ENGINEERING
Faculty of Industrial Technology
Institut Teknologi Sepuluh November
Surabaya 2015

STRATEGI PENYERANGAN NPC BERBASIS *RULE* PADA *GAME REAL TIME STRATEGY*

TUGAS AKHIR

Diajukan untuk Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Teknik
Pada
Bidang Studi Teknik Komputer dan Telematika
Jurusan Teknik Elektro
Institut Teknologi Sepuluh Nopember

Menyetujui:

Dosen Pembimbing I	Dosen Pembimbing II
Mochamad Hariadi, ST.,M.Sc.,Ph.D.	Dr. Supeno Mardi Susiki N, ST., MT.
NIP. 19691209 1997031 002	NIP. 19700313 1995121 001

SURABAYA
JULI, 2015

STRATEGI PENYERANGAN NPC BERBASIS *RULE* PADA *GAME REAL TIME STRATEGY*

Nama : I Putu Widya Wiranata
Pembimbing : 1. Mochamad Hariadi, ST., M.Sc., Ph.D.
2. Dr. Supeno Mardi S N, ST., MT.

ABSTRAK

Real Time Strategy adalah salah satu genre permainan. Pemain bermain sebagai salah satu kubu dalam perang. Pemain mengatur strategi untuk dapat mengalahkan musuh. Permainan ini menerapkan sistem *real-time*. Pada *game* tersebut ada karakter yang dinamakan NPC. NPC (*Non-Player Character*) adalah karakter yang berjalan tanpa adanya kontrol dari *player*. Biasanya NPC menyerang terus hingga tewas. Pada kenyataannya hal tersebut kurang realistis. Ketika pasukan dalam keaadan terdesak maka akan kembali ke markas. Pada penelitian ini diajukan strategi penyerangan yang dapat menyerang musuh dan kembali ke markas menggunakan *rule-based*. Percobaan dilakukan dengan membandingkan NPC musuh dengan sistem *battle* poin dan NPC pemain tanpa sistem *battle* poin. Dari pengujian didapatkan hasil, NPC musuh dengan battle poin memiliki tingkat kemenangan sebesar 60% dibandingkan dengan NPC pemain tanpa *battle* poin hanya 40%.

Kata kunci : *Real Time Strategy*, NPC, *rule based*

Halaman ini sengaja dikosongkan

ATTACK STRATEGY OF NPC RULE BASED ON REAL TIME STRATEGY GAME

Name **: I Putu Widya Wiranata**
Supervisor **: 1. Mochamad Hariadi, ST., M.Sc., Ph.D.**
2. Dr. Supeno Mardi S N, ST., MT.

ABSTRACT

Real Time Strategy is one of the game genres. Player play as one the side of the war. Player use a strategy to defeat the enemy. The game implement real-time system. A character called NPC is the game. NPC (Non-Player Character) is a character that run without any player control. Usually NPC attack enemy till death. It is less realistic. When troops in a state of urgency then the troops will return to headquarters. This research proposed a strategy to attack the enemy and return to headquarters using a rule. Experiments in this study has been done by testing NPC enemies using battle points while NPC players without battle points. The result at the experiment shows that the NPC enemy with battle points have winning rate of 60% compared to the NPC players without battle points of 40% only.

Keywords :*Real Time Strategy, rule based, NPC*

Halaman ini sengaja dikosongkan

DAFTAR ISI

DAFTAR TABEL

Halaman ini sengaja dikosongkan

DAFTAR GAMBAR

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Pada saat ini terdapat banyak sekali jenis game yang telah beredar di masyarakat. Beberapa jenis game tersebut adalah *action, adventure, shooter, action-adventure, role-playing game, simulation,* dan *real time strategy*. *Real Time Strategy* adalah salah satu genre permainan yang sering diasosiasikan sebagai *war game*, secara umum pemain bermain sebagai salah satu kubu dalam sebuah peperangan, membangun strategi yang efektif untuk dapat mengalahkan musuh secepat dan sebaik mungkin. Permainan ini menerapkan elemen *real-time*, sehingga pemain akan mengatur strategi pada saat perang dimulai. Dalam permainan ini strategi dan *war* adalah bagian utama, dimana membangun strategi yang efektif serta menyelesaikan perang secepat mungkin adalah kunci kemenangan permainan. Pada game tersebut ada karakter yang dinamakan NPC. NPC (*Non-Player Character*) adalah karakter pada game dimana karakter tersebut berjalan tanpa adanya kontrol dari *player*. Kebanyakan NPC pada game bersifat reaktif (menunggu aksi dari player) seperti NPC penjual barang, NPC pemberi *quest*, dan NPC musuh pada game RPG. Tetapi pada game RTS untuk melakukan penyerangan kepada pemain atau *player* diperlukan NPC yang tidak hanya bereaksi berdasarkan aksi yang dilakukan oleh *player* tetapi juga beraksi secara mandiri sesuai dengan *rules* yang telah ditetapkan kepada NPC tersebut. Pada penelitian sebelumnya telah dibuat game RTS, yaitu Lume Wars dimana Lume Wars ini merupakan 2D RTS *game*. Pada penelitian sebelumnya strategi penyerangan yang dilakukan hanya sebatas menyerang, bertahan dan berkelompok. Game RTS menyajikan permainan yang memerlukan pengambilan tindakan yang tepat sehingga strategi yang terbentuk dapat berjalan pada skenario yang telah dirancang. Oleh karena itu diperlukan NPC berbasis aturan (*Rule Based System*) untuk melakukan strategi penyerangan agar lebih terstruktur, dimana NPC dapat menyerang musuh, bertahan dari serangan musuh dan menghindar ketika menghadapi musuh lebih kuat. Ketika akan melakukan penyerangan NPC melakukan *patrolling* dan *sensoring* untuk mendapatkan posisi dari *player*. NPC

mempunyai *Finite State Machine* (FSM) untuk menentukan perilaku penyerangan yang harus diambil, baik berdasarkan kondisi saat ini (*current state*) maupun berikutnya (*next state*) sesuai dengan aturan-aturan yang telah ditetapkan. Implementasi dari penelitian ini adalah membuat strategi penyerangan pada game RTS menggunakan berbasis aturan (*Rule Based System*).

1.2 Permasalahan

Masalah yang dihadapi untuk ditemukan solusinya melalui tugas akhir ini adalah sebagai berikut :

1. Strategi penyerangan npc yang terdapat pada *game* RTS biasanya sekali menyerang maka npc tersebut akan menyerang terus hingga tewas, akan tetapi pada kenyataannya hal tersebut kurang realistis karena jika npc tersebut diibaratkan sebagai suatu pasukan maka pada saat kondisi terdesak pasukan tersebut akan memilih mundur. Oleh karena itu pada penelitian ini diajukan strategi penyerangan NPC yang dapat menyerang markas musuh dan kembali ke markasnya berbasis aturan (*Rule Based System*).

1.3 Tujuan

Tujuan yang diharapkan tercapai setelah selesainya tugas akhir ini adalah sebagai berikut :

1. Menghasilkan NPC yang dapat melakukan penyerangan secara autonomous dan terstruktur berbasis aturan.

2. Membandingkan NPC yang hanya dapat maju dan maju mundur untuk melakukan penyerangan.

1.4 Batasan Masalah

Permasalahan yang dibahas dalam Tugas Akhir ini memiliki beberapa batasan, diantaranya sebagai berikut:

1. Penelusuran masalah hanya sampai mengatur stategi penyerangan pada NPC.

2. Pengujian skenario dilakukan menggunakan NPC melawan NPC.

1.5 Metodologi

Metodologi yang digunakan dalam penyusunan Tugas Akhir ini digambarkan pada gambar 1.1.

Gambar 1.1 Metodelogi

1. Perancangan Skenario Penyerangan dan Parameter yang akan Digunakan

Sebelum proses untuk membuat strategi penyerangan, terlebih dahulu dilakukan perancangan skenario penyerangan dan parameter yang akan digunakan untuk melakukan penyerangan pada *game* RTS.

2. Perancangan Aturan Strategi Penyerangan

Perancangan strategi menyerang untuk NPC dengan menggunakan aturan-aturan.

3. Penerapan Strategi Penyerangan NPC pada Skenario

Strategi penyerangan yang telah dibuat dengan menggabungkan aturan-aturan yang telah dirancang kemudian diterapkan sesuai dengan skenario pada *game*.

4. Pengujian Performansi dan pengambilan Data

Proses selanjutnya yaitu pengukuran performansi dan pengambilan data dapat dilakukan. Pada tahapan ini, dilakukan pengumpulan data untuk berbagai kondisi sesuai dengan parameter yang telah ditentukan untuk pengujian. Tujuan akhirnya adalah menentukan seberapa baik aturan-aturan yang telah diterapkan pada strategi penyerangan.

5. Analisa dan Kesimpulan

Analisa data dan pengambilan kesimpulan mengacu pada data hasil pengujian.

6. Dokumentasi dan Laporan

Tahapan ini meliputi pembuatan laporan tugas akhir dan penulisan jurnal ilmiah. Pembuatan laporan tugas akhir untuk beberapa bagian dilakukan bersesuaian dengan pengerjaan tahapan-tahapan diatas. Sedangkan jurnal ilmiah dilakukan setelah laporan tugas akhir selesai.

1.6 Sistematika Penulisan

Sistematika penulisan laporan tugas akhir ini dibagi dalam lima bab, masing-masing bab diuraikan sebagai berikut:

1. BAB 1 PENDAHULUAN
   Bab ini menjelaskan tentang pendahuluan dari tugas akhir yang meliputi latar belakang, permasalahan, tujuan, batasan masalah dan sistematika penulisan.
2. BAB 2 TEORI PENUNJANG
   Bagian ini menjelaskan tentang teori-teori penunjang yang digunakan sebagai dasar dalam penelitian tugas akhir ini.
3. BAB 3 DESAIN DAN IMPLEMENTASI
   Bab ini berisi tentang penjelasan desain dan perancangan strategi menyerang pada NPC sesuai dengan skenario dengan berbasis *rule*.
4. BAB 4 PENGUJIAN DAN ANALISA
   Bab ini merupakan pemaparan dan analisa hasil pengujian implementasi strategi penyerangan NPC pada *game* Fantasy Chronicles.
5. BAB 5 PENUTUP
   Bab ini berisi kesimpulan dari penelitian berdasarkan data pengujian, serta saran yang membangun untuk pengembangan selanjutnya.

Halaman ini sengaja dikosongkan

BAB 2
TEORI PENUNJANG

Pada penelitian ini didukung oleh teori penunjang sebagai bahan acuan dan referensi agar penelitian lebih terarah.

2.1 *Real Time Strategy*

Real Time Strategy (RTS) adalah genre permainan komputer di mana pemain mengendalikan tertentu, biasanya militer, *aset* (*soldier*, kendaraan, struktur) yang pemain dapat memanipulasi untuk mencapai kemenangan atas musuh. Sehingga pada game RTS diperlukan kemampuan selain untuk mengatur pasukan dan kondisinya juga perlu untuk membaca dan mengantisipasi pergerakan dari musuh atau lawan, dengan begitu pemain dapat memenangkan game tersebut. RTS telah menjadi salah satu genre video *game* yang paling populer sejak sekitar tahun 90-an memasuki *mainstream* dengan merilis terobosan Dune II (dirilis pada 1992) yang mendefinisikan genre dengan memperkenalkan banyak komponen dasar yang hampir setiap game RTS yang modern termasuk. Dalam RTS permainan khas, adalah mungkin untuk membuat unit (baik berupa unit untuk bertempur maupun unit untuk melakukan produksi) dan struktur tambahan selama permainan. Menciptakan aset tambahan mengharuskan pemain untuk menghabiskan sumber daya. Sumber daya ini biasanya dalam bentuk beberapa mata uang moneter atau bahan berharga (emas, kayu, batu, minyak mentah dan bahan-bahan makanan pokok). Sumber daya ini dapat diperoleh selama permainan juga, paling sering dengan membuat unit yang bisa panen bahan berharga. Dalam kebanyakan game RTS ada cara lain dari sumber terakumulasi, seperti mengontrol poin khusus pada peta permainan atau dengan mengendalikan struktur dikhususkan untuk ini (misalnya tambang minyak, tambang emas, lahan hutan kayu dan sumber makanan baik yang berada di darat maupun yang ada di sungai atau laut) [8]. *Real time strategy* dibedakan dari *turn-based strategy* dimana pada game RTS tidak mengenal giliran karena bersifat *real time*. Setiap pemain dapat mengatur atau memerintah pasukannya dalam waktu apapun. Dalam RTS, tema permainan dapat berupa sejarah (misalnya seri Age of Empire), fantasi (misalnya Warcraft) dan fiksi ilmiah (misalnya Star Wars). Gambar 2.1 merupakan contoh salah satu permainan *real time strategy*[6].

Gambar 2.1 Permainan *Real Time Strategy* Starcraft [6]

2.2 *Non Player Character*

Non-Player Characters (NPC) atau disebut juga agen adalah suatu entitas dalam game yang tidak dikendalikan secara langsung oleh pemain. NPC dikendalikan secara otomatis oleh komputer. NPC bisa berupa teman, musuh atau netral. NPC diinginkan dapat berperilaku cerdas layaknya manusia. Dia bisa mengindera lingkungan, berpikir, memilih aksi lalu bertindak sebagai respon atas perubahan pada lingkungannya. Untuk dapat memperoleh perilaku cerdas dari NPC digunakan kecerdasan buatan atau *Artificial Intelligence* (AI). Penggunaan AI pada NPC dilakukan dengan pemberian algoritma khusus sesuai dengan perilaku cerdas yang diharapkan. Pada adegan (scene) pertempuran dari permainan computer, prajurit pemain dan prajurit musuh merupakan contoh dari NPC. Perilaku seorang prajurit dalam medan pertempuran bervariasi mulai dari mengikuti pimpinan, menghindari halangan, berlari, berjalan, menjauhi musuh, bertarung, membantu teman, dan lainnya[5].

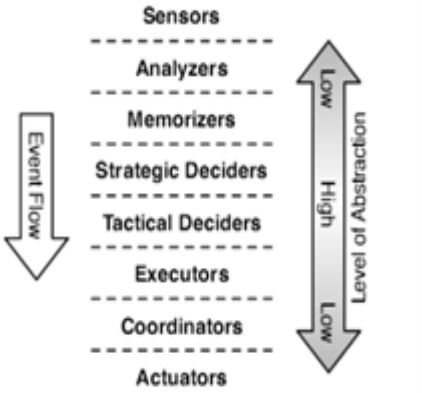
2.3 *Artificial Intelligence* pada Permainan RTS

Pengembangkan sistem cerdas untuk *game* RTS adalah masalah yang kompleks. Karena dalam RTS selain kendala intrinsik dari permainan, seperti pengamatan parsial, ada dua jenis keputusan yaitu

1.*micro-managemen* : bertanggung jawab atas perilaku unit
2.*macro-management* : bertanggung jawab atas pembangunan

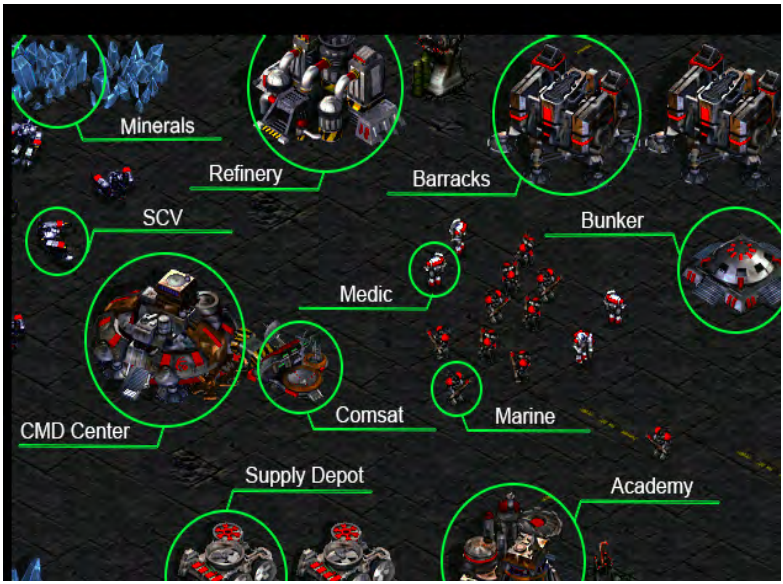
Struktur dari managemen sumber daya dan strategi melibatkan tugas alokasi. Mungkin aspek ini membuat jenis *game* kurang diteliti sebagai obyek yang saling bergantian, meskipun jumlahnya besar dan terdapat banyak tantangan yang terbuka seperti: perencanaan *real-time*, pengambilan keputusan di bawah ketidakpastian dari *par-tially* domain yang diamati, belajar dari pengalaman atau pengamatan, lawan modeling, penalaran spasial dan temporal, navigasi, re-manajemen sumber dan kolaborasi.

Teknik yang paling sering digunakan dalam pengembangan *Artificial Intelligence* di permainan *Real Time Strategy* adalah: perubahan strategi pada permainan RTS, model bayessian, *potential fields*, *evolutionary algorithms* dan masih banyak lagi metode-metode yang digunakan dalam meningkatkan kecerdasaan pada *game* RTS[4].Untuk arsitektur model AI pada *game* dapat dilihat pada gambar 2.2.



Gambar 2.2 Arsitektur AI pada *game* [1]

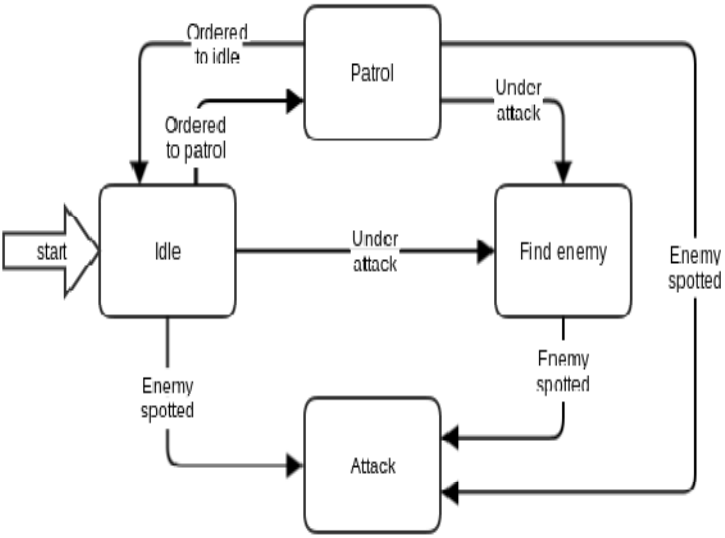
Untuk *screenshoot* dari contoh permainan RTS Starcraft dalam berbagai karakteristik seperti macam-macam bangunan dan pasukan perang dapat dilihat pada gambar 2.3.



Gambar 2.3 Permainan Starcraft dengan fungsi dan karakteristik yang berbeda-beda [4]

2.4 *Finite State Machine*

Menurut Rabin, salah satu teknik pengembangan *Artificial Intelligence* dalam permainan adalah *Finite State Machine* (FSM). Popularitasnya besar dan dapat dijelaskan oleh kualitas yang dimiliki FSMs misalnya mereka dapat diterapkan untuk banyak masalah AI yang berbeda dan mudah untuk memahami dan melaksanakan. Sebuah *Finite State Machine* (FSM) terdiri dari satu set *state* terbatas di mana hanya satu *state* pada suatu waktu dapat aktif. Biasanya satu *state* ditandai sebagai *state* mulai dan memutuskan kemana perhitungan dimulai. Untuk dapat berpindah dari satu *state* ke yang lain, transisi harus didefinisikan terlebih dahulu. Kondisi dan peristiwa yang terkait dengan transisi dan digunakan untuk menjelaskan ketika transisi seharusnya memicu. Gambar 2.4 menampilkan contoh dari FSM[3].



Gambar 2.4 Contoh FSM AI pada *game* [3]

2.5 *Knowledge Based System*

DSS (*decission support systems*) merupakan alat bantu untuk memecahkan persoalan-persoalan yang kurang terstruktur dengan baik. KBS (*knowledge-based system* atau *expert system*) merupakan alat bantu untuk persoalan yang kurang terstruktur dengan baik. KBS (*system* atau *expert system*) merupakan alat bantu untuk memecahkan persoalan yang mengikuti cara kerja pakar (*expert*). Penggabungan antara KBS dan DSS yang disebut dengan KB-DSS memberi suatu alternative baru untuk memecahkan persoalan. *Knowledge Based System* adalah suatu sistem yang menggunakan set (*knowledge*) yang dikodekan ke bahasa mesin untuk dapat menyimpulkan dan melakukan suatu tugas. *Knowledge Based System* digunakan untuk dapat membantu manusia dalam menyelesaikan masalah yang dihadapi dengan berdasarkan atas pengetahuan yang telah diprogramkan ke system tersebut. Untuk hal inilah maka digunakan *knowledge based system* dalam memecahkan masalah yang berhubungan AI[2].

2.6 *Rule Based System*

Sistem berbasis aturan (*Rule Based System*) adalah suatu program komputer yang memproses informasi yang terdapat di dalam working memory dengan sekumpulan aturan yang terdapat di dalam basis pengetahuan menggunakan mesin inferensi untuk menghasilkan informasi baru.

Sebuah Rule-Based System dapat dibentuk dengan menggunakan sebuah assertions set, yang secara kolektif membentuk *working memory*, dan sebuah rule set yang menentukan aksi pada assertions set. RBS secara relatif adalah model sederhana yang bisa diadaptasi ke banyak masalah. Namun, jika ada terlalu banyak peraturan, pemeliharaan sistem akan rumit dan terdapat banyak *failure* dalam kerjanya.

Untuk membuat sistem berbasis aturan, anda harus memiliki :

Sekumpulan fakta yang mewakili *working memory*. Ini dapat berupa suatu keadaan yang relevan dengan keadaan awal sistem bekerja.

Sekumpulan aturan. Aturan ini mencakup setiap tindakan yang harus diambil dalam ruang lingkup permasalahan yang dibutuhkan.

Kondisi yang menentukan bahwa solusi telah ditemukan atau tidak (*none exist*). Hal ini berguna untuk menghindari *looping* yang tidak akan pernah berakhir.

Teori sistem berbasis aturan ini menggunakan tekhnik yang sederhana, yang dimulai dengan dasar aturan yang berisi semua pengetahuan dari permasalahan yang dihadapi yang kemudian dikodekan ke dalam aturan IF-THEN dan sebuah tempat penyimpanan (basis data) yang mengandung data, pernyataan dan informasi awal. Sistem akan memeriksa semua aturan kondisi (IF) yang menentukan subset, set konflik yang ada. Jika ditemukan, maka sistem akan melakukan kondisi THEN. Perulangan atau *looping* ini akan terus berlanjut hingga salah satu atau dua kondisi bertemu, jika aturan tidak diketemukan maka sistem tersebut harus keluar dari perulangan (*terminate*).

2.7 Unity3D

Unity 3D adalah sebuah *game engine* yang berbasis *cross-platform*. Unity dapat digunakan untuk membuat sebuah game yang bisa digunakan pada perangkat komputer, ponsel pintar android, iPhone, ps3.

Unity adalah sebuah sebuah *tool* yang terintegrasi untuk membuat *game*, arsitektur bangunan dan simulasi. Unity bisa untuk

games PC dan *games Online*. Untuk *games Online* diperlukan sebuah plugin, yaitu Unity Web Player, sama halnya dengan Flash Player pada Browser.

Unity tidak dirancang untuk proses desain atau modelling, dikarenakan unity bukan *tool* untuk mendesain. Jika ingin mendesain, pergunakan 3D editor lain seperti 3dsmax atau Blender. Banyak hal yang bisa dilakukan dengan unity, ada fitur audio reverb zone, particle effect, dan Sky Box untuk menambahkan efek langit pada *game*.

Fitur scripting yang disediakan, mendukung 3 bahasa pemrograman, JavaScript, C#, dan Boo. Flexible and EasyMoving, rotating, dan scaling objects hanya perlu sebaris kode. Begitu juga dengan Duplicating, removing, dan changing properties. Visual Properties Variables yang di definisikan dengan scripts ditampilkan pada Editor. Bisa digeser, di drag and drop, bisa memilih warna dengan color picker. Berbasis .NET. Artinya penjalanan program dilakukan dengan Open Source .NET platform, Mono.

Gambar 2.5 Tampilan *game engine* Unity3D

Pada setiap project Unity terdapat sebuah Assets folder. Isi dari Assets folder ditampilkan dalam bentuk panel project dalam editor unity. Assets folder adalah tempat untuk menyimpan semua komponen dari game seperti tingkatan game (level scenes), scripts, 3D models, teksture.

Untuk menambahkan assets ke dalam project, cukup dengan menarik (drag) file yang ingin ditambahkan ke dalam panel project. Atau dengan memilih menu *Assets->Import New Asset*. Untuk membuat scene baru, gunakan tombol Control-N (pada keyboard). Untuk menyimpan scene yang sedang aktif, gunakan Control-S (pada keyboard).

Panel *hierarchy* menampung semua *GameObject* yang terdapat di *Scene* yang sedang aktif. Beberapa dari *GameObject* tersebut berhubungan langsung ke asset seperti objek 3D. Objek yang terdapat pada *hierarchy* dapat di seleksi dan dihapus. Jika objek dihapus atau ditambahkan pada scene, maka objek tersebut juga akan hilang atau muncul pada *hierarchy*.

2.8 Fantasy Chronicles

Fantasy Chronicles merupakan permainan 3D dengan bergenre *Real Time Strategy* yang dikembangkan oleh tim beranggotakan 5 orang mhasiswa dari lintas jalur teknik komputer dan telematika. Permainan ini memiliki latar belakang terjadi pertempuran antara 2 ras utama yaitu *White Warrio*r (NPC *player*) dan *Black Warrior* (NPC musuh). Kedua ras tersebut memiliki 4 kelompok yang sama yang terdiri dari penyerangan, pertahanan, bangunan, dan pekerja. Dalam permainan ini kedua ras saling berhadapan untuk memperebutkan kekuasaan dan saling menghancurkan. Di kelompok penyerangan ras *White Warrior* memiliki 6 macam (NPC *player*) yang berperang menggunakan pedang dan sihir api. Sedangkan di ras *Black Warrior* memiliki 6 macam (NPC musuh) yang berperang menggunakan senjata (pedang, kapak ,tongkat ,gada) dan sihir api hitam. Permainan *Fantasy Chronicles* 3D ini meliputi 5 bagian penting yaitu perilaku NPC, *flocking*, strategi penyerangan, *scoring* dan *gameplay*. Kelima unsur tersebut nantinya akan dijadikan satu dan diujicobakan ke *game* yang sebenarnya kemudian akan dilakukan analisa terhadap permainan tersebut. Tampilan awal dari permainan *Fantasy Chronicles* 3D dapat dilihat pada gambar 2.6.

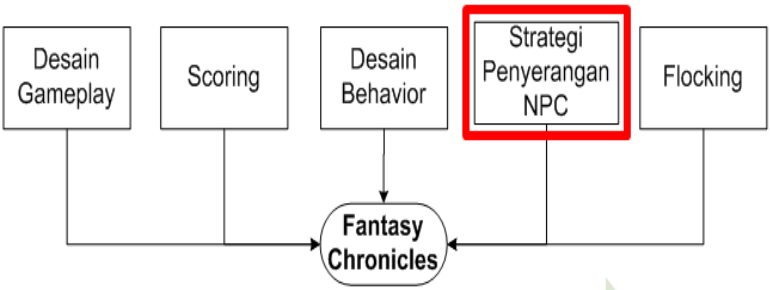
Gambar 2.6 Tampilan awal permainan *Fantasy Chronicles* 3D

Halaman ini sengaja dikosongkan

BAB 3
DESAIN DAN IMPLEMENTASI

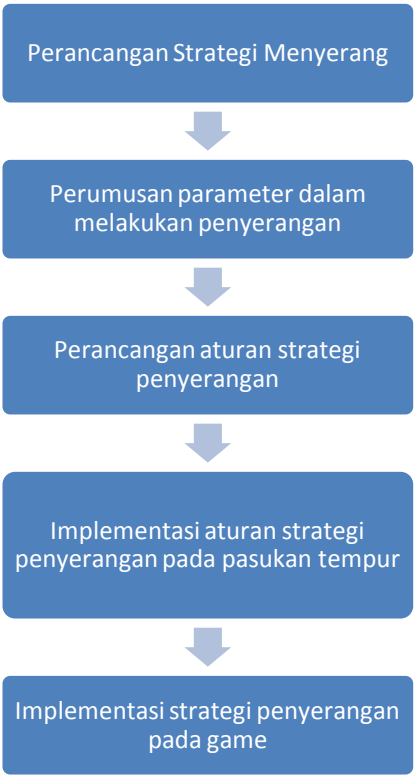
3.1 Rancangan Sistem

Fantasy Chronicles merupakan penelitian yang dikerjakan oleh lima orang mahasiswa jurusan Teknik Elektro ITS. Fantasy Chronicles merupakan permainan 3D bergenre *real time strategy* yang terdiri dari beberapa bagian yang digabungkan menjadi satu kesatuan *game*. Rancangan sistem permainan Fantasy Chronicles secara keseluruhan dapat dilihat pada gambar 3.1.



Gambar 3.1 Rancangan sistem *game Fantasy Chronicles*

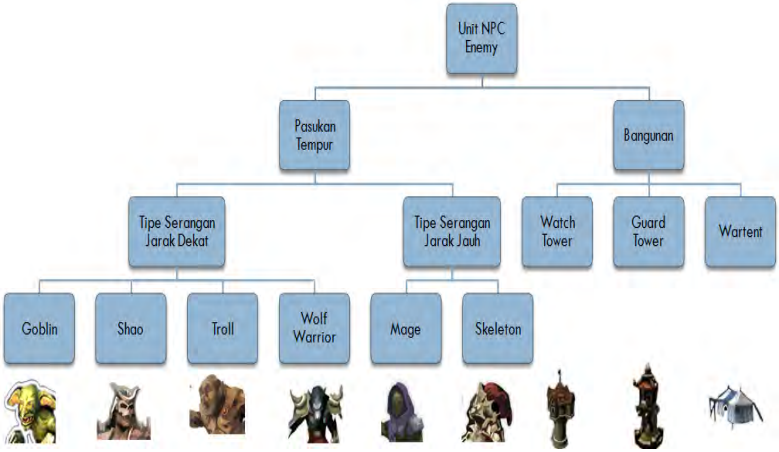
Dari rancangan sistem pada gambar 3.1 terdapat 5 bagian utama pada permainan Fantasy Chronicles, yaitu desain *gameplay*, *scoring*, desain *behaviour*, strategi penyerangan npc, dan *flocking*. Dari kelima bagian tersebut yang dibahas pada penelitian ini adalah strategi penyerangan npc. Desain dari penelitian dari strategi penyerangan tersebut meliputi perancangan macam-macam strategi menyerang yang digunakan, perumusan parameter yang digunakan dalam melakukan penyerangan, perancangan aturan-aturan yang digunakan pada strategi penyerangan, mengimplementasikan aturan-aturan tersebut pada pasukan tempur dalam melakukan penyerangan terhadap musuh pada *game*. Setelah itu akan dilakukan beberapa percobaan untuk menguji keefektifan strategi penyerangan yang digunakan dalam menghadapi musuh pada permainan Fantasy Chronicles. Desain perancangan strategi penyerangan npc pada permainan Fantasy Chronicles secara garis besar dapat dilihat pada gambar 3.2.



Gambar 3.2 Desain perancangan strategi penyerangan NPC

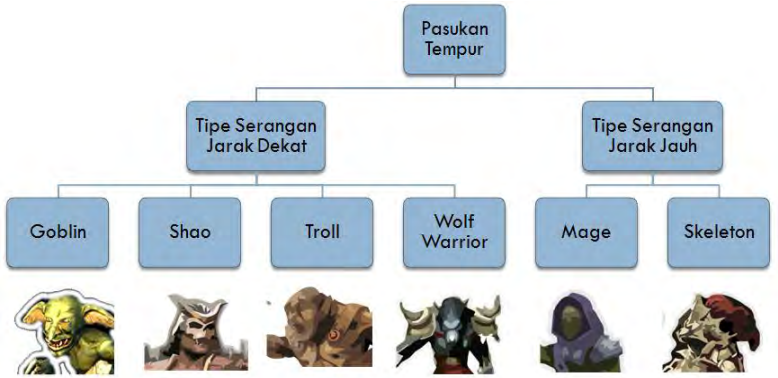
3.1.1 Unit NPC Musuh

Unit NPC musuh yang digunakan pada permainan *Fantasy Chronicles* ini dibagi menjadi 2 bagian, yaitu unit bangunan dan unit pasukan tempur. Dimana unit bangunan tersebut digunakan untuk menciptakan pasukan-pasukan tempur, sedangkan unit pasukan tempur digunakan untuk menyerang pasukan musuh dan menghancurkan bangunan musuh dengan tujuan untuk memenangkan permainan. Unit pasukan tempur terdiri dari 2 tipe serangan, yaitu tipe pasukan dengan serangan jarak dekat dan tipe pasukan dengan serangan jarak jauh. Berikut ini adalah desain unit NPC musuh pada permainan *Fantasy Chronicles* dapat dilihat pada gambar 3.3.



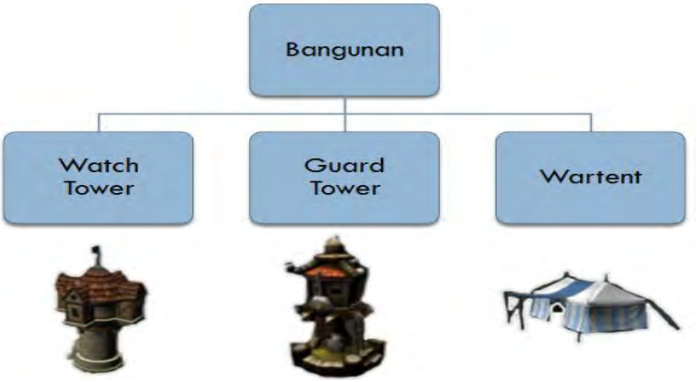
Gambar 3.3 Unit NPC musuh

Pasukan tempur pada *game* Fantasy Chronicles ini untuk tipe serangan jarak dekat menggunakan senjata seperti pedang, kapak, clurit, dan golok. Sedangkan tipe serangan jarak jauh menggunakan senjata tongkat sihir, untuk menyerang lawannya pasukan ini mengeluarkan sihir bola api yang akan melukai lawannya jika terkena bola api tersebut. Pada *game* ini dibuat lebih dari satu NPC dengan jenis NPC yang berbeda-beda agar strategi penyerangan yang dilakukan menjadi lebih menarik[1]. Desain pasukan tempur dapat dilihat pada gambar 3.4.



Gambar 3.4 Unit pasukan tempur

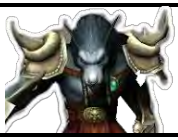
Unit bangunan digunakan untuk membuat pasukan-pasukan tempur. Dimana nantinya pasukan-pasukan tempur ini yang akan melindungi bangunan tersebut. Desain bangunan dapat dilihat pada gambar 3.5.



Gambar 3.5 Unit Bangunan

Daftar karakter NPC musuh yang digunakan pada permainan *Fantasy Chronicles* ini dapat dilihat pada tabel 3.1.

Tabel 3.1 Daftar karakter pada NPC musuh

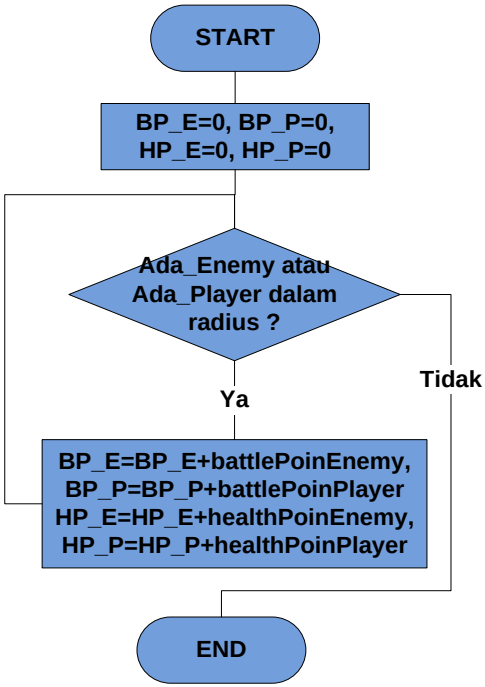
Gambar	Nama	Tipe	Karakteristik
	*Wolf Warrior*	NPC	Menyerang
	*Goblin*	NPC	Menyerang
	*Mage*	NPC	Menyerang

Gambar	Nama	Tipe	Karakteristik
	*Skeleton*	NPC	Menyerang
	*Troll*	NPC	Menyerang
	*Shao*	NPC	Menyerang
	*Watchtower*	NPC	Membuat Pasukan Tempur
	*Guardtower*	NPC	Membuat Pasukan Tempur
	*Wartent*	NPC	Membuat pasukan Tempur

3.1.2 Sistem Perang Game Fantasy Chronicles

Pada bagian ini akan dijelaskan mengenai sistem perang pada permainan *Fantasy Chronicles*. Pada permainan ini pertempuran menggunakan sistem *battle* poin. Dimana untuk setiap unit ketika melakukan pertarungan akan menghitung *battle* poin dan *health* poinnya begitu juga dengan *battle* poin dan *health* poin musuhnya, jika *battle*

poin dan *health* poinnya lebih besar atau sama dengan *battle* poin dan *health* poin musuhnya maka unit tersebut akan menyerang sebaliknya jika *battle* poin dan *health* poinnya kurang dari *battle* poin dan *health* poin musuhnya maka unit tersebut akan kabur atau menghindar. Jika terdapat karakter yang sepihak yang masuk dalam radius maka *battle* poin dan *health* poin karakter tersebut akan dijumlahkan. Flowchart perhitungan *battle* poin dan *health* poin pada permainan ini dapat dilihat pada gambar 3.6.

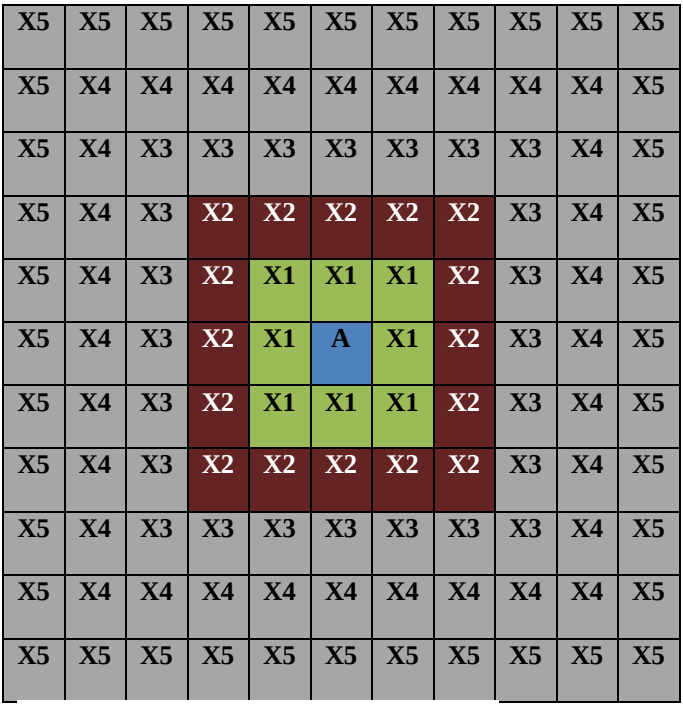


Gambar 3.6 Flowchart perhitungan battle poin dan health poin

3.1.3 Jangkauan Serangan Karakter

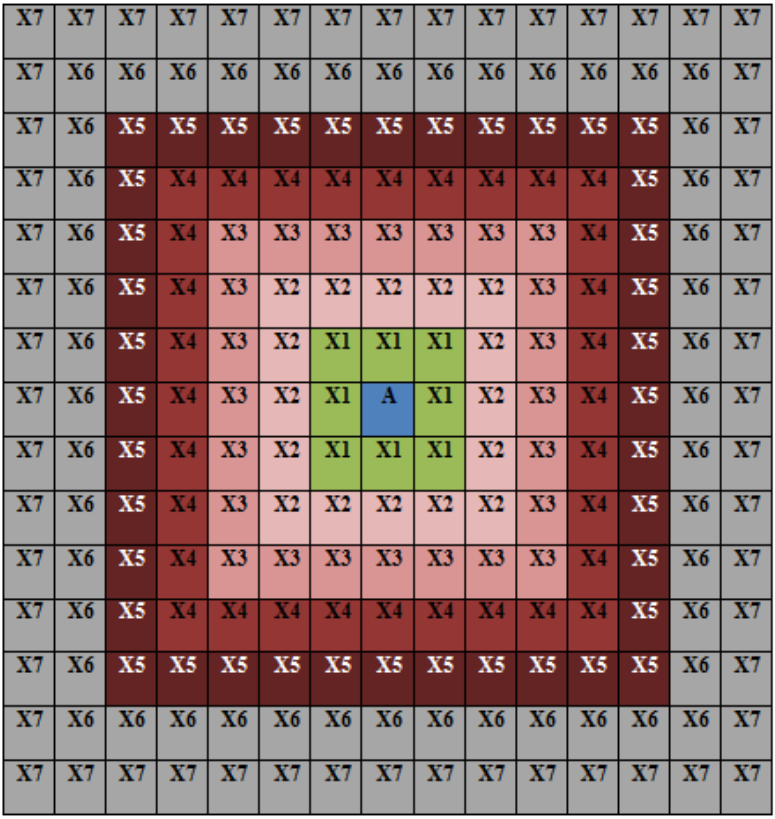
Pada bagian ini akan dibahas mengenai jangkauan serangan dari tiap karakter, dimana jangkauan ini berguna untuk menghitung *battle* poin dari pasukan pemain dan pasukan musuh juga untuk melakukan serangan terhadap karakter musuh yang ada di dalam

jangkauan serangan karakter tersebut. Pada karakter NPC musuh jangkauan serangan dibagi menjadi 2, yaitu jangkauan serangan jarak dekat dan jangkauan serangan jarak jauh. Untuk jangkauan serangan jarak dekat diimplementasikan pada karakter goblin, shao, wolf warrior dan troll. Jangkauan serangan jarak dekat dapat dilihat pada gambar 3.7.



Gambar 3.7 Jangkauan serangan jarak dekat

A adalah simbol posisi dari karakter, X1-X2 adalah jangkauan untuk melakukan serangan jarak dekat kepada karakter musuh yang berada pada area tersebut dan X1-X5 adalah jangkauan untuk menghitung *battle* poin tiap unit karakter yang berada pada area tersebut. Untuk jangkauan serangan jarak jauh diimplementasikan pada karakter mage dan skeleton. Jangkauan serangan jarak jauh dapat dilihat pada gambar 3.8.



Gambar 3.8 Jangkauan serangan jarak jauh

A adalah simbol posisi dari karakter, X1-X5 adalah jangkauan untuk melakukan serangan jarak jauh kepada karakter musuh yang berada pada area tersebut dan X1-X7 adalah jangkauan untuk menghitung *battle* poin tiap unit karakter yang berada pada area tersebut.

3.1.4 Parameter Strategi Penyerangan

Pada permainan Fantasy Chronicles setiap karakter memiliki karakteristik yang berbeda-beda. Karakteristik tersebut yang akan

menjadi parameter masukan untuk strategi penyerangan NPC. Karakter akan digunakan pada tiap *scene* sesuai dengan skenario pada *scene* tersebut. Berikut ini adalah tabel karakteristik dari pasukan tempur NPC dapat dilihat pada tabel 3.2.

Tabel 3.2 Daftar atribut pada NPC musuh

NAMA	HEALTH POIN	BATTLE POIN	RADIUS AREA	SYARAT UNIT	
				KAYU	POIN
Goblin	50	4	25	10	2
Shao	50	4	25	10	2
Mage	50	6	100	20	6
Wolf Warrior	50	8	25	12	4
Troll	50	14	25	50	12
Skeleton	50	15	100	40	10

Dari tabel 3.2 terdapat 4 parameter yaitu battle poin, radius area, emas dan poin. Berikut ini adalah penjelasan dari parameter-parameter tersebut :

1. Battle poin

   Parameter *battle* poin digunakan untuk mengukur tingkat kekuatan karakter NPC terhadap karakter musuh. Jika *battle* poin karakter NPC lebih besar atau sama dengan karakter musuh maka karakter tersebut dapat menyerang musuh tetapi jika *battle* poin karakter NPC lebih kecil dari karakter musuh maka karakter tersebut akan menghindar atau kabur dari karakter musuh.

2. Radius area/jangkauan area

   Parameter jangkauan area karakter digunakan untuk menghitung jumlah battle poin timnya dan musuhnya dalam jangkauan area karakter dan juga untuk melakukan serangan kepada musuh yang masuk ke dalam jangkauan area dari karakter.

3. Kayu

Parameter ini merupakan syarat yang digunakan untuk memproduksi pasukan penyerangan. Setiap pasukan memiliki syarat kayunya masing-masing. Semakin besar syarat yang dibutuhkan maka semakin kuat karakter tersebut.

4. Poin

Parameter ini merupakan syarat yang digunakan untuk memproduksi pasukan penyerangan. Setiap pasukan memiliki syarat poinnya masing-masing. Semakin besar syarat yang dibutuhkan maka semakin kuat karakter tersebut.

5. Health poin

Parameter *health* poin digunakan untuk mengukur tingkat kekuatan karakter NPC terhadap karakter musuh. Jika *health* poin karakter NPC lebih besar atau sama dengan karakter musuh maka karakter tersebut dapat menyerang musuh tetapi jika *health* poin karakter NPC lebih kecil dari karakter musuh maka karakter tersebut akan menghindar atau kabur dari karakter musuh. Parameter kesehatan juga digunakan untuk pengambilan tindakan dari karakter NPC tersebut, jika HP > 50% maka NPC akan menyerang musuh jika HP < 50% maka NPC akan kembali ke markas untuk melakukan penyembuhan.

3.2 Perancangan Aturan Strategi Penyerangan

Pada bagian ini akan dijelaskan mengenai desain perancangan aturan strategi penyerangan yang berfungsi untuk mengatur penyerangan yang dilakukan oleh NPC pada game Fantasy Chronicles. Aturan-aturan ini akan menentukan keputusan yang akan diambil oleh npc tersebut. FSM utama dari strategi menyerang NPC ini dibagi menjadi 3 bagian, yaitu menyerang markas musuh dan melindungi markas dan kembali ke markas.

Keterangan :

BP_E (%) = persentase dari selisih *battle* poin musuh dan *battle* poin pemain dibandingkan dengan *battle* poin musuh.

$$\mathrm{BP_E}(\%) = \frac{bp_{musu\ h} - bp_{pemain}}{bp_{musu\ h}} * 100\%$$

BP_P (%) = persentase dari selisih *battle* poin pemain dan *battle* poin musuh dibandingkan dengan *battle* poin pemain.

$$\mathrm{BP_P}(\%) = \frac{bp_{pemain} - bp_{musu\,h}}{bp_{pemain}} * 100\%$$

HP_E (%) = persentase dari selisih *health* poin musuh dan *health* poin pemain dibandingkan dengan *health* poin musuh.

$$\mathrm{HP_E}(\%) = \frac{hp_{musu\,h} - hp_{pemain}}{hp_{musu\,h}} * 100\%$$

HP_P (%) = persentase dari selisih *health* poin pemain dan *health* poin musuh dibandingkan dengan *health* poin pemain.

$$\mathrm{HP_P}(\%) = \frac{hp_{pemain} - hp_{musu\,h}}{hp_{pemain}} * 100\%$$

Berikut ini adalah aturan dalam menyerang musuh dengan membandingkan kondisi *battle* poin dan *health* poin dari npc musuh dan pemain. Pada aturan ini keputusan yang diambil dalam melakukan penyerangan lebih mengacu pada kondisi *battle* poin dibandingkan kondisi *health* poin. Aturan dalam menyerang musuh dapat dilihat pada tabel 3.3.

Tabel 3.3 Daftar *rule* dengan menggunakan operator AND

KONDISI		**KEPUTUSAN**
Battle Poin	Health Poin	Menyerang Musuh
BP_E < 50%	HP_E < 50%	Ya
BP_E < 50%	HP_E >= 50%	Ya
BP_E >= 50%	HP_E < 50%	Ya
BP_E >= 50%	HP_E >= 50%	Ya
BP_E < 50%	HP_P < 50%	Ya
BP_E < 50%	HP_P >= 50%	Tidak
BP_E >= 50%	HP_P < 50%	Ya
BP_E >= 50%	HP_P >= 50%	Ya
BP_P < 50%	HP_E < 50%	Tidak

KONDISI		KEPUTUSAN
BP_P < 50%	HP_E >= 50%	Ya
BP_P >= 50%	HP_E < 50%	Tidak
BP_P >= 50%	HP_E >= 50%	Tidak
BP_P < 50%	HP_P < 50%	Tidak
BP_P < 50%	HP_P >= 50%	Tidak
BP_P >= 50%	HP_P < 50%	Tidak
BP_P >= 50%	HP_P >= 50%	Tidak

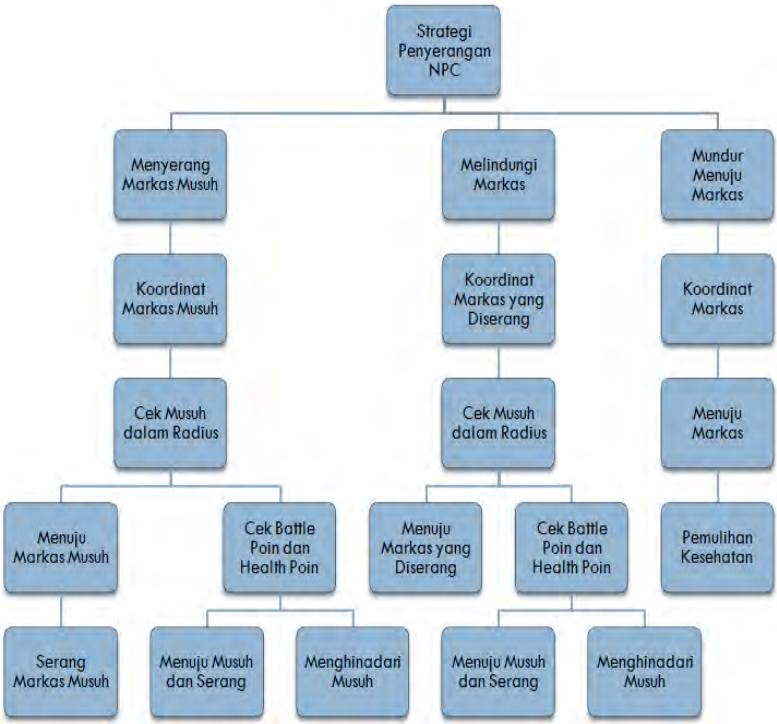
Berikut ini adalah aturan dalam menyerang musuh dengan membandingkan kondisi *battle* poin dan *health* poin dari npc musuh dan pemain. Pada aturan ini keputusan yang diambil dalam melakukan penyerangan lebih mengacu pada kondisi *health* poin dibandingkan kondisi *battle* poin. Aturan dalam menyerang musuh dapat dilihat pada tabel 3.4.

Tabel 3.4 Daftar *rule* dengan menggunakan operator AND

KONDISI		KEPUTUSAN
Battle Poin	Health Poin	Menyerang Musuh
BP_E < 50%	HP_E < 50%	Ya
BP_E < 50%	HP_E >= 50%	Ya
BP_E >= 50%	HP_E < 50%	Ya
BP_E >= 50%	HP_E >= 50%	Ya
BP_E < 50%	HP_P < 50%	Tidak
BP_E < 50%	HP_P >= 50%	Tidak
BP_E >= 50%	HP_P < 50%	Ya
BP_E >= 50%	HP_P >= 50%	Tidak
BP_P < 50%	HP_E < 50%	Ya
BP_P < 50%	HP_E >= 50%	Ya
BP_P >= 50%	HP_E < 50%	Tidak
BP_P >= 50%	HP_E >= 50%	Ya
BP_P < 50%	HP_P < 50%	Tidak
BP_P < 50%	HP_P >= 50%	Tidak
BP_P >= 50%	HP_P < 50%	Tidak
BP_P >= 50%	HP_P >= 50%	Tidak

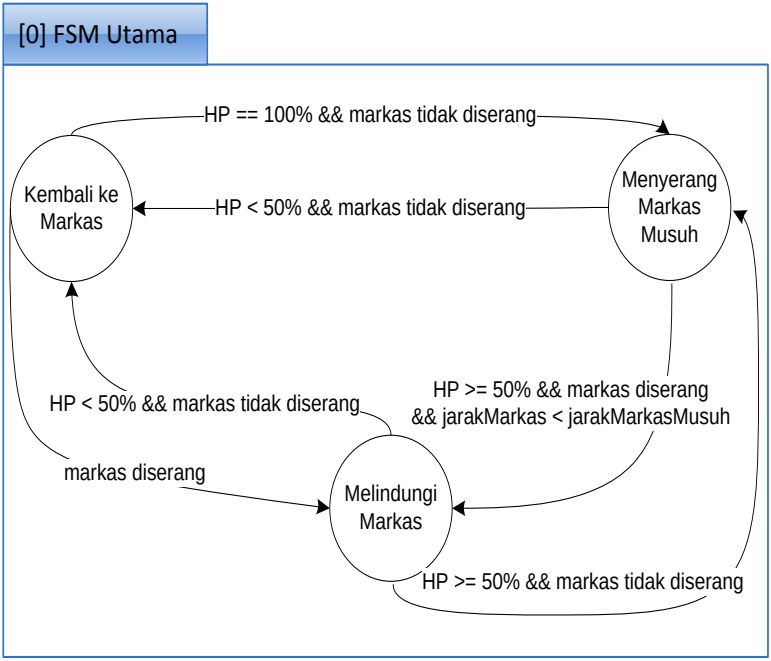
3.2.1 FSM Utama

Pada FSM utama ini strategi penyerangan NPC dibagi menjadi 3 strategi penyerangan, yaitu menyerang markas musuh, melindungi markas dan kembali menuju ke markas. Saat menyerang markas musuh diperlukan koordinat markas musuh yang terdekat dari NPC tersebut. Kemudian melakukan pengecekan apakah terdapat musuh disekitar NPC tersebut, jika ada maka akan dicek *battle* poin dan *health* poinnya, jika tidak maka akan melanjutkan serangan ke markas musuh. Saat melindungi markas diperlukan koordinat markas yang diserang oleh musuh. Kemudian menuju ke markas tersebut dan melindunginya dari serangan musuh. Saat kembali ke markas adalah ketika kondisi NPC tersebut tidak memungkinkan untuk melanjutkan serangan. Strategi penyerangan pada permainan ini dapat dilihat pada gambar 3.9.



Gambar 3.9 Strategi penyerangan *game Fantasy Chronicles*

Untuk FSM utama dari strategi menyerang NPC ini dapat dilihat pada gambar 3.10.



Gambar 3.10 FSM utama strategi penyerangan

FSM utama ini terdiri dari 3 *state*, yaitu :

1. Kembali ke Markas

   Kondisi ini dicapai ketika kondisi kesehatan karakter kurang dari 50%. Karakter tersebut akan kembali ke markas untuk melakukan pemulihan kesehatan. Kondisi ini memiliki FSM lagi didalamnya.

2. Menyerang Markas Musuh

   Kondisi ini dicapai ketika karakter tersebut baru dibuat, karakter tersebut akan langsung menyerang markas musuh jika markas dari karakter tersebut tidak diserang oleh musuh. Kondisi ini memiliki FSM lagi didalamnya.

3. Melindungi Markas

   Kondisi ini dicapai apabila markas dari karakter tersebut diserang oleh musuh, sehingga karakter tersebut akan kembali menuju markas yang diserang. Setelah mencapai markas, karakter tersebut akan menyerang musuh yang berada di sekitar markas tersebut. Kondisi ini memiliki FSM lagi didalamnya.

Pada FSM utama tersebut perpindahan dari *state* satu ke *state* lainnya diatur berdasarkan beberapa parameter yang telah ditetapkan. Parameter tersebut adalah health poin, kondisi markasnya, jarak markasnya dengannya bila dibandingkan dengan jarak markas musuh dengannya. Berikut adalah aturan dari FSM utama dapat dilihat pada tabel 3.5.

Tabel 3.5 Tabel aturan pada FSM utama

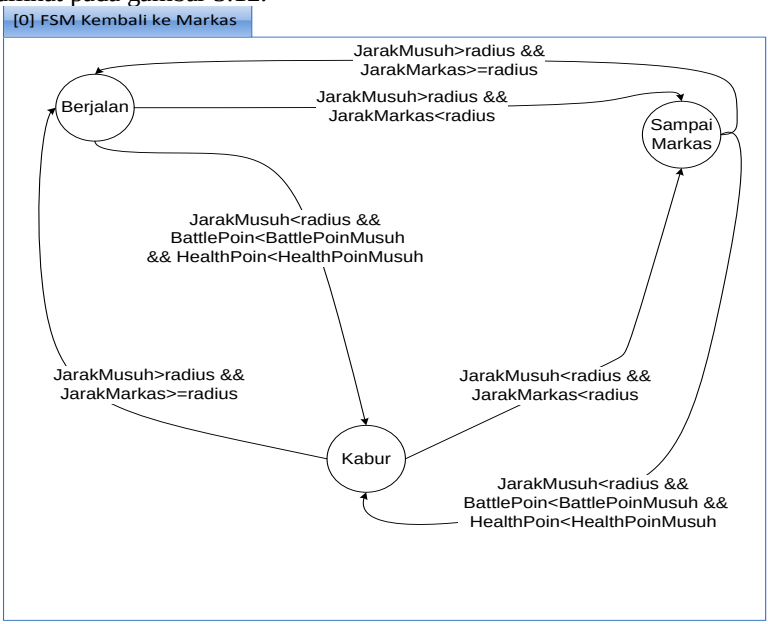
KONDISI				**KEPUTUSAN**
State Saat Ini	**Heath Poin**	**Markas Diserang**	**Jarak Markas Dibandingkan Jarak Markas Musuh**	
Kembali ke markas	100%	Tidak	-	Menyerang markas musuh
Kembali ke markas	-	Ya	-	Melindungi markas
Menyerang markas musuh	<50%	Tidak	-	Kembali ke markas
Menyerang markas musuh	>=50%	Ya	Lebih dekat	Melindungi markas
Melindungi markas	<50%	Tidak	-	Kembali ke markas
Melindungi markas	>=50%	Tidak	-	Menyerang markas musuh

3.2.2 Tahapan Kembali ke Markas

Untuk tahapan kembali ke markas pada permainan *Fantasy Chronicles* ini dapat dilihat pada gambar 3.11.

Gambar 3.11 Tahapan kembali ke markas

Untuk FSM dari state kembali ke markas pada game ini dapat dilihat pada gambar 3.12.



Gambar 3.12 FSM dari state kembali ke markas

Pada FSM ini terdapat dari 3 *state*, yaitu :

1. Berjalan

   Kondisi ini dicapai ketika karakter baru dibuat dan tidak menjalankan kondisi apapun, tidak terdapat musuh dalam jangkauan karakter tersebut.

2. Sampai Markas

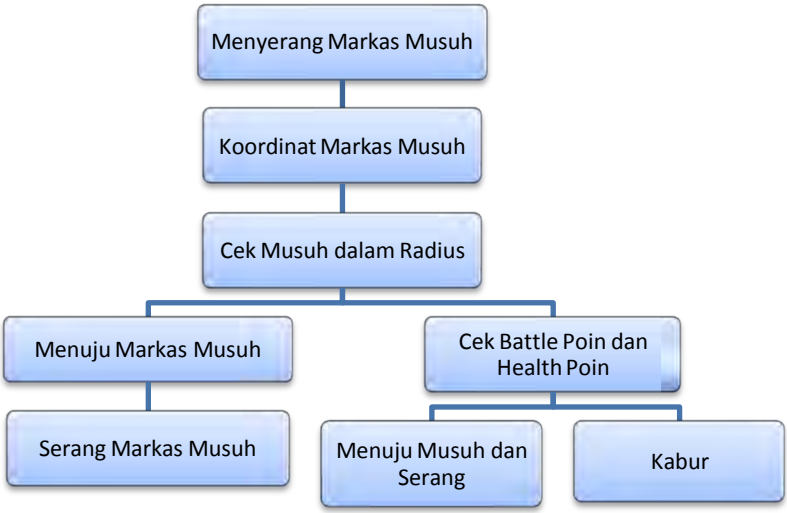
   Kondisi ini adalah ketika karakter telah sampai pada markas sehingga karakter tersebut dapat melakukan pemulihan kesehatan.

3. Kabur

   Kondisi ini dicapai apabila dalam perjalanan menuju ke markas terdapat musuh yang menghadang sehingga perlu untuk dihindari.

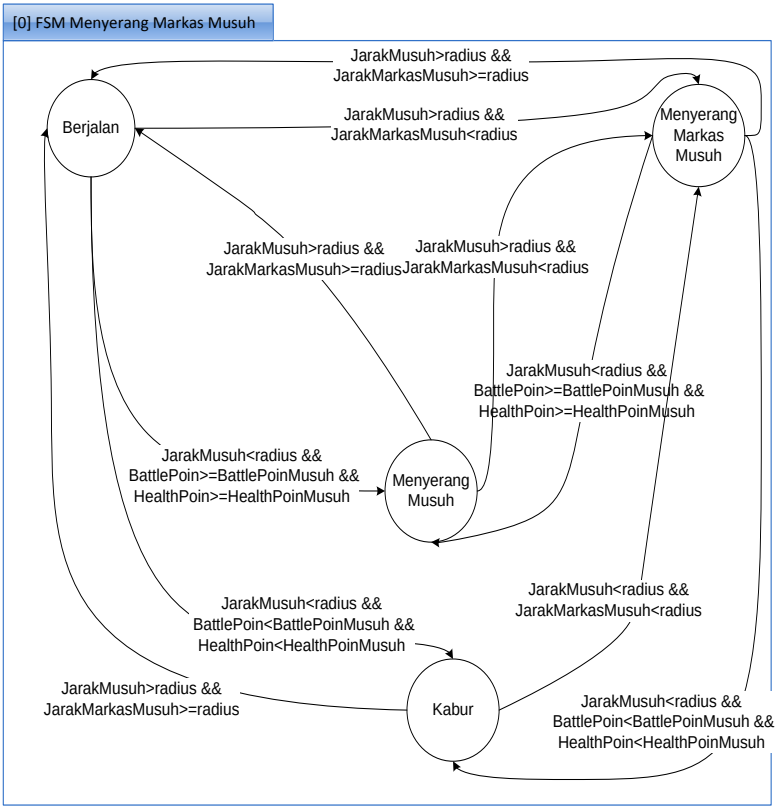
3.2.3 Tahapan Menyerang

Untuk tahapan state menyerang markas musuh pada permainan ini dapat dilihat pada gambar 3.13.



Gambar 3.13 Tahapan menyerang markas musuh

Untuk FSM dari state menyerang markas musuh pada *game* ini dapat dilihat pada gambar 3.14.



Gambar 3.14 FSM dari state menyerang markas musuh

Pada FSM ini terdapat dari 4 *state*, yaitu :

1. Berjalan

   Kondisi ini adalah ketika karakter terpilih untuk melakukan penyerangan menuju ke markas musuh, jarak markas musuh lebih besar dari jangkauan karakter tersebut dan tidak terdapat musuh dalam jangkauan karakter tersebut.
2. Menyerang Markas Musuh

   Kondisi ini adalah ketika karakter terpilih untuk melakukan penyerangan, markas musuh berada dalam

jangkauan karakter tersebut dan tidak terdapat musuh dalam jangkauan karakter tersebut.

3. Menyerang Musuh

   Kondisi ini dicapai apabila terdapat musuh dalam jangkauan karakter dan total *battle point* dari karakter lebih besar daripada total *battle point* dari musuh.

4. Kabur

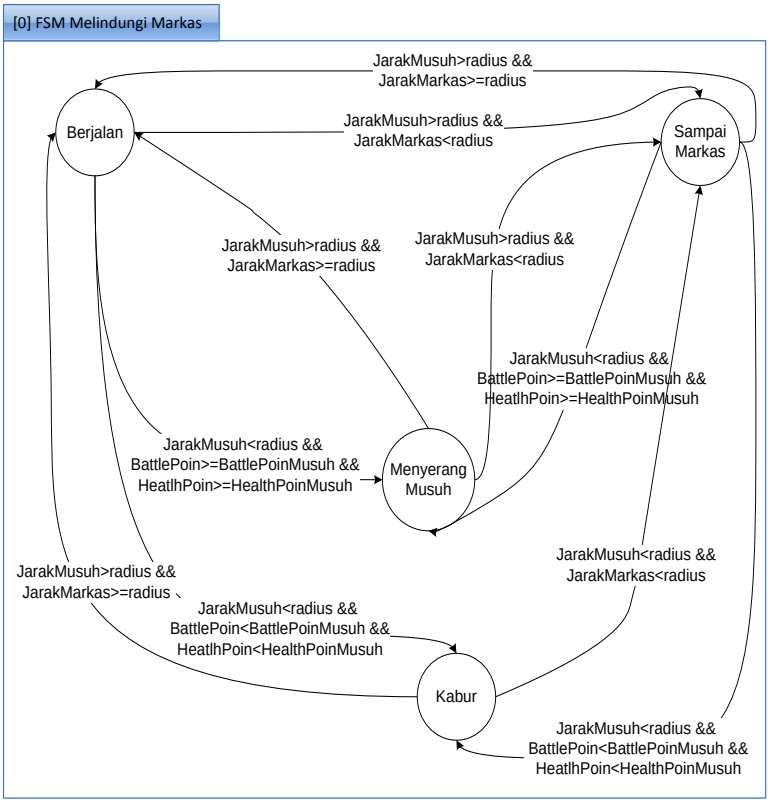
   Kondisi ini dicapai apabila terdapat musuh dalam jangkauan karakter dan total *battle point* dari karakter lebih kecil daripada total *battle point* dari musuh.

3.2.4 Tahapan Melindungi Markas

Untuk hirarki state menyerang markas musuh pada game ini dapat dilihat pada gambar 3.15.

Gambar 3.15 Tahapan melindungi markas

Untuk FSM dari state melindungi markas pada permainan ini dapat dilihat pada gambar 3.16.



Gambar 3.16 FSM dari state melindungi markas

Pada FSM ini terdapat dari 4 *state*, yaitu :

1. Berjalan

   Kondisi ini adalah ketika karakter melakukan pertahan menuju ke markas yang diserang oleh musuh, jarak markas lebih besar dari jangkauan karakter tersebut dan tidak terdapat musuh dalam jangkauan karakter tersebut.

2. Sampai Markas

   Kondisi ini adalah ketika karakter melakukan pertahanan dan telah sampai pada markas yang diserang oleh musuh dan tidak terdapat musuh dalam jangkauan karakter tersebut.

3. Menyerang Musuh

   Kondisi ini dicapai apabila terdapat musuh dalam jangkauan karakter dan total *battle point* dan *health point* dari karakter lebih besar daripada total *battle point* dan *health point* dari musuh.

4. Kabur

   Kondisi ini dicapai apabila terdapat musuh dalam jangkauan karakter dan total *battle point* dan *health point* dari karakter lebih kecil daripada total *battle point* dan *health point* dari musuh.

3.3 Implementasi Aplikasi

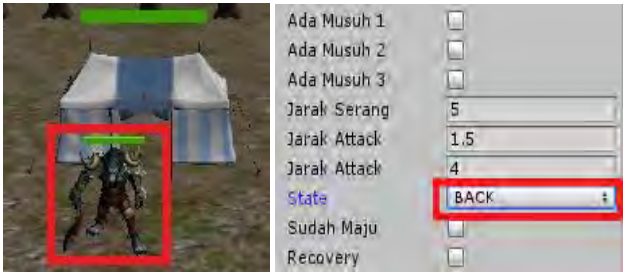
Pada bagian ini akan dijelaskan mengenai penggabungan skenario dan strategi penyerangan pada NPC serta implementasinya pada permainan *Fantasy Chronicles.*

3.3.1 Implementasi Strategi Penyerangan pada NPC

Terdapat 5 state pada strategi penyerangan NPC, yaitu:

1. Kembali ke Markas

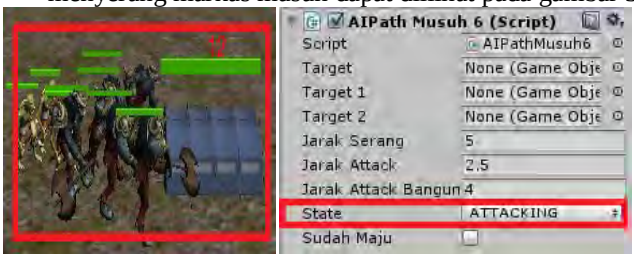
   Karakter berada pada kondisi kembali ke markas ketika kesehatan dari karakter tersebut kurang dari 50%. Pada saat mundur ke markas karakter memprioritaskan markas yang memiliki tingkat penyembuhan paling besar, yaitu maras asal dari karakter tersebut. Tetapi jika markas tersebut dalam kondisi diserang oleh musuh maka karakter tersebut akan kembali ke markas yang paling dekat dengannya. Berikut ini adalah implementasi dari state siaga dapat dlilihat pada gambar 3.17.



Gambar 3.17 Implementasi state kembali ke markas

2. Menyerang Markas Musuh

Karakter berada pada kondisi menyerang markas musuh ketika karakter tersebut diberi perintah untuk menyerang markas musuh. Jika markas musuh berada dalam jangkauan karakter tersebut maka markas musuh akan diserang. Berikut ini adalah implementasi dari state menyerang markas musuh dapat dlilihat pada gambar 3.18.



Gambar 3.18 Implementasi state menyerang markas musuh

3. Bertahan Melindungi Markas

Karakter berada pada kondisi bertahan melindungi markas ketika karakter tersebut diberi perintah untuk bertahan melindungi markas. Jika di sekitar markas terdapat karakter musuh maka, musuh tersebut akan diserang. Berikut ini adalah implementasi state bertahan melindungi markas musuh dapat dilihat pada gambar 3.19.

Gambar 3.19 Implementasi state melindungi markas

4. Perang

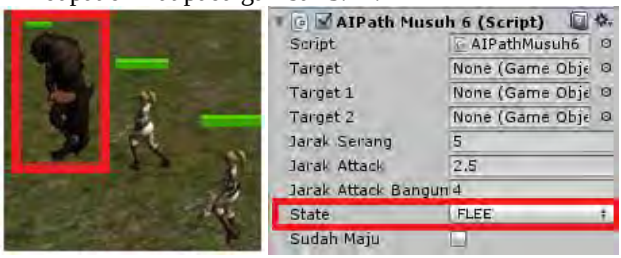
Karakter beraada pada kondisi berperang ketika terdapat musuh yang berada di dalam jangkauan karakter tersebut dan *battle* poin dari karakter tersebut lebih besar

daripada musuh. Berikut ini adalah implementasi dari state perang dapat dilihat pada gambar 3.20.

Gambar 3.20 Implementasi state perang

5. Kabur

Karakter berada pada kondisi kabur ketika terdapat musuh yang berada di dalam jangkauan karakter tersebut dan *battle* poin dari karakter tersebut lebih kecil daripada musuh. Berikut ini adalah implementasi dari state kabur dapat dilihat pada gambar 3.21.



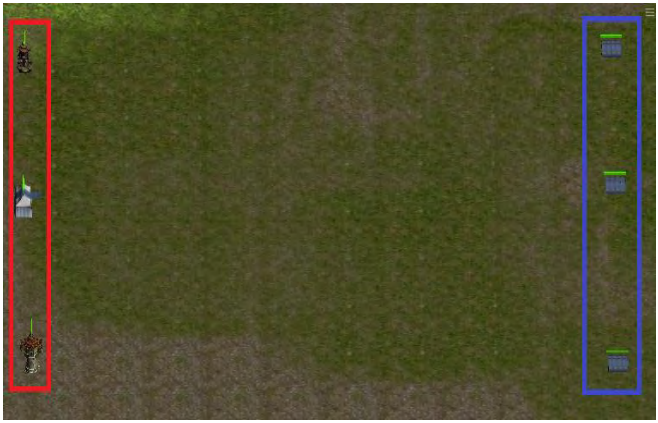
Gambar 3.21 Implementasi state kabur

3.3.2 Implementasi Strategi Penyerangan pada Skenario

Strategi penyerangan yang telah dibuat akan diimplementasikan pada 2 skenario dalam permainan *Fantasy Chronicles*, yaitu skenario pada map dengan area terbuka dan skenario pada map dengan area labirin.

1. Skenario pada Area Terbuka

Pada skenario ini area yang digunakan adalah area terbuka tanpa halangan dimana markas musuh tepat berada di depan markas npc. Sehingga pada saat pasukan NPC musuh dan pemain berada di tengah maka akan terjadi perang. Area perang pada skenario ini dapat dilihat pada gambar 3.22.



Gambar 3.22 Map area perang

Pada daerah kotak merah adalah markas dari npc musuh sedangkan pada daerah kotak biru adalah markas dari npc pemain. Pada skenario ini terdapat 3 jenis pasukan tempur yang dibuat, yaitu troll, goblin dan mage. Setiap jenis pasukan tempur akan dibuat sebanyak 1 kelompok yaitu berjumlah 5 pasukan, sehingga total pasukan sekali dibuat sebanyak 15. Pasukan yang dibuat tersebut akan menyerang markas musuh jika markasnya tidak diserang dan tidak terdapat musuh yang berada dalam radiusnya. Komposisi pasukan pada skenario ini dapat dilihat pada gambar 3.23.

Gambar 3.23 Komposisi pasukan tempur NPC musuh

Pada skenario ini pasukan musuh dan pasukan pemain akan bertemu di tengah area. Pasukan NPC musuh dan pemain bertemu di tengah area dapat dilihat pada gambar 3.24.

Gambar 3.24 NPC pemain dan musuh bertermpur

Ketika kondisi kesehatan npc musuh berada di bawah 50% maka npc tersebut akan mundur ke markasnya untuk melakukan penyembuhan. Dari 3 markas yang ada akan diprioritaskan untuk kembali ke markas asal dimana karakter tersebut dibuat karena penyembuhan yang diterima akan lebih banyak dari pada markas yang lainnya, akan tetapi jika markas asal tersebut sedang diserang oleh musuh maka karakter tersebut tidak akan kembali ke markas tersebut sehingga akan dipilih markas dengan jarak yang terdekat dengannya. Pasukan NPC musuh kembali ke markas dapat dilihat pada gambar 3.25.

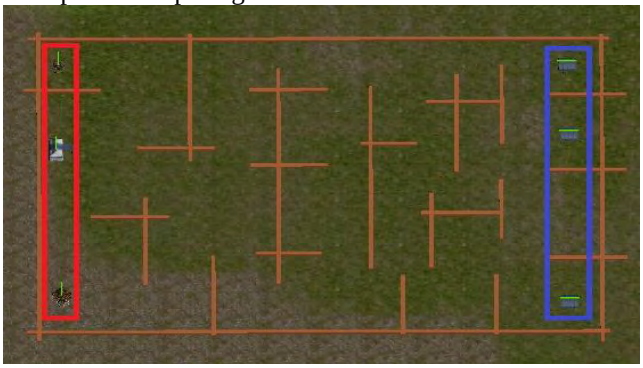
Gambar 3.25 NPC musuh kembali ke markas

Pada skenario ini ketika npc pemain menyerang markas musuh maka npc musuh akan bertahan untuk menghentikan serangan npc pemain tersebut dengan syarat jika jarak markas yang diserang oleh npc pemain kurang dari jarak markas yang akan diserang oleh npc musuh tersebut. Jika jaraknya lebih jauh maka npc musuh akan fokus untuk menyerang markas pemain. Npc musuh bertahan dari serangan npc pemain dapat dilihat pada gambar 3.26.

Gambar 3.26 NPC musuh dan bertahan dari serangan npc pemain

2. Skenario pada Area Labirin

Pada skenario ini area yang digunakan adalah area labirin dengan halangan yang berliku-liku. Area perang pada skenario ini dapat dilihat pada gambar 3.27.



Gambar 3.27 Map area perang

Pada daerah kotak merah adalah markas dari npc musuh sedangkan pada daerah kotak biru adalah markas dari npc pemain. Pada skenario ini terdapat 3 jenis pasukan tempur yang dibuat, yaitu troll, goblin dan mage. Setiap jenis pasukan tempur akan dibuat sebanyak 1 kelompok yaitu berjumlah 5 pasukan, sehingga total pasukan sekali dibuat sebanyak 15. Pasukan yang dibuat tersebut akan menyerang markas musuh jika markasnya tidak diserang dan tidak terdapat musuh yang berada dalam radiusnya. Komposisi pasukan pada skenario ini dapat dilihat pada gambar 3.28.

Gambar 3.28 Komposisi pasukan tempur npc musuh

Pada skenario ini pasukan musuh dan pasukan pemain akan melewati jalur yang berliku-liku untuk mencapai markas lawannya. Pasukan NPC musuh melewati area dengan halangan yang berliku-liku dapat dilihat pada gambar 3.29.

Gambar 3.29 NPC musuh melewati halangan

Ketika kondisi kesehatan npc musuh berada di bawah 50% maka npc tersebut akan mundur ke markasnya untuk melakukan penyembuhan. Dari 3 markas yang ada akan diprioritaskan untuk kembali ke markas asal dimana karakter tersebut dibuat karena penyembuhan yang diterima pada markas asal tersebut akan lebih banyak dari pada markas yang lainnya, akan tetapi jika markas asal tersebut sedang diserang oleh musuh maka karakter tersebut tidak akan kembali ke markas tersebut sehingga akan dipilih markas dengan jarak yang berada paling dekat dengannya. Pada skenario ini tidak banyak npc yang bisa kembali ke markas karena medan jalan yang berliku-liku sehingga menyulitkan karakter tersebut untuk menuju markasnya. Pasukan NPC musuh kembali ke markas dapat dilihat pada gambar 3.30.

Gambar 3.30 NPC musuh kembali ke markas

Pada skenario ini ketika npc pemain menyerang markas musuh maka npc musuh akan bertahan untuk menghentikan serangan npc pemain tersebut dengan syarat jika jarak markas yang diserang oleh npc pemain tersebut kurang dari jarak markas yang akan diserang oleh npc musuh tersebut. Jika jaraknya lebih jauh maka npc musuh akan memilih tetap fokus untuk menyerang markas pemain yang sedang dituju. NPC musuh bertahan dari serangan npc pemain dapat dilihat pada gambar 3.31.

Gambar 3.31 NPC musuh bertahan dari serangan NPC pemain

Ketika tidak terdapat pasukan pemain yang berada dalam radius dan markas tidak sedang diserang maka npc musuh akan menyerang markas dari pemain untuk memengangkan permainan. Npc musuh menyerang markas pemain dapat dilihat pada gambar 3.32.

Gambar 3.32 NPC musuh menyerang markas pemain

Halaman ini sengaja dikosongkan

BAB 4
PENGUJIAN DAN ANALISA

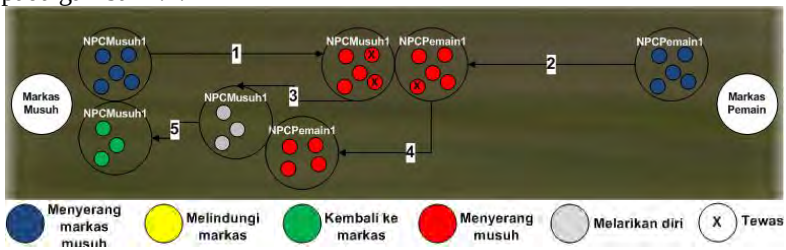
Dalam bab ini membahas mengenai pengujian dari sistem yang telah direalisasikan. Hal ini bertujuan untuk mengetahui apakah sistem yang di rencanakan telah bekerja dengan baik dan sesuai harapan. Pengujian pada penelitian ini dilakukan dengan menguji apakah strategi penyerangan yang telah dirancang efektif apabila diterapakan pada *game* Fantasy Chronicles.

4.1 Pengujian Strategi Penyerangan

Pengujian ini bertujuan untuk mengetahui sejauh mana strategi penyerangan untuk NPC ini dapat berjalan pada permainan Fantasy Chronicles. Pengujian ini dilakukan dengan menghitung lama waktu permainan yang berlangsung pada setiap skenario yang telah dirancang dengan menggunakan perbandingan dari NPC yang memiliki strategi menyerang, bertahan, kembali ke markas dengan NPC yang hanya memiliki strategi menyerang dan bertahan.

4.1.1 Percobaan NPC Pemain melawan NPC Musuh

Penyerangan yang dilakukan oleh NPC pemain dan NPC musuh terdiri dari 3 kelompok yang akan menyerang secara bergantian dimana setiap kelompok terdiri dari 5 anggota. Berikut ini adalah simulasi percobaan NPC pemain melawan NPC musuh dapat dilihat pada gambar 4.1.



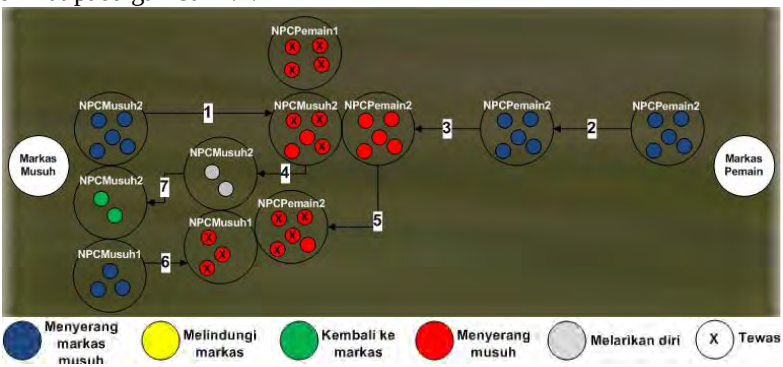
Gambar 4.1 Pasukan NPC musuh1 melawan pasukan NPC pemain1

Penjelasan dari gambar 4.1 adalah sebagai berikut:

1. Kelompok NPC musuh1 dalam perjalanan menyerang markas musuh dan bertemu NPC pemain1 di tengah.

2. Kelompok NPC pemain1 dalam perjalanan menyerang markas musuh dan bertemu NPC musuh1 di tengah.
3. Saat bertempur NPC musuh1 sisa 3 sehingga kondisi *battle* poin tidak mencukupi maka NPC musuh1 kembali ke markas.
4. Saat bertempur NPC pemain1 sisa 4 dan mengejar NPC musuh1
5. NPC musuh1 sampai di markas dan melakukan pemulihan kesehatan.

Selanjutnya penyerangan dilakukan oleh NPC kelompok2 dapat dilihat pada gambar 4.2.



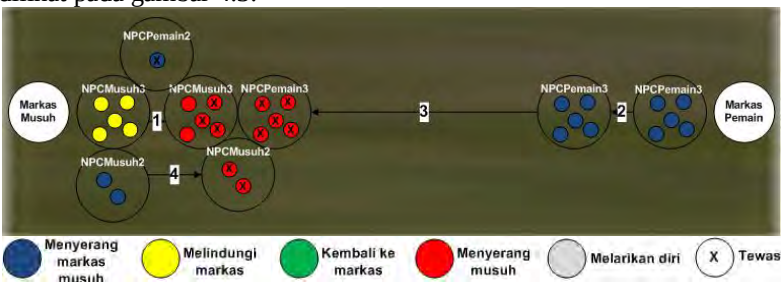
Gambar 4.2 Pasukan NPC musuh2 melawan pasukan NPC pemain2

Penjelasan dari gambar 4.2 adalah sebagai berikut:

1. Kelompok NPC musuh2 menghadang kelompok NPC pemain1.
2. Kelompok NPC pemain2 dalam perjalanan menyerang markas musuh.
3. Kelompok NPC pemain2 menyerang kelompok NPC musuh2.
4. Saat bertempur NPC musuh2 sisa 2 sehingga kondisi *battle* poin tidak mencukupi maka NPC musuh kembali ke markas.
5. NPC pemain2 mengejar NPC musuh2 tetapi dihadang oleh NPC musuh1.

6. NPC musuh1 setelah melakukan pemulihan kesehatan kemudian menghadang NPC pemain2.
7. NPC musuh2 sampai di markas dan melakukan pemulihan kesehatan.

Selanjutnya penyerangan dilakukan oleh kelompok 3 dapat dilihat pada gambar 4.3.

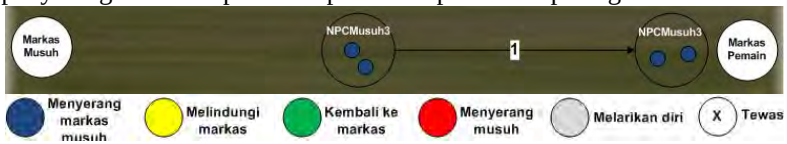


Gambar 4.3 Pasukan NPC musuh3 melawan pasukan NPC pemain3

Penjelasan dari gambar 4.3 adalah sebagai berikut:

1. NPC musuh3 melindungi markas yang diserang oleh NPC pemain2.
2. Kelompok NPC pemain3 dalam perjalanan menyerang markas musuh.
3. NPC pemain3 menyerang NPC musuh3.
4. NPC musuh2 setelah melakukan pemulihan kesehatan kemudian ikut menyerang NPC pemain3.

Setelah pertempuran tersebut NPC musuh3 tersisa 2 dan NPC pemain tidak tersisa, kemudian NPC musuh3 tersebut melakukan penyerangan terhadap markas pemain dapat dilihat pada gambar 4.4.



Gambar 4.4 Pasukan NPC musuh3 menyerang markas pemain

Penjelasan dari gambar 4.4 adalah sebagai berikut:

1. NPC musuh3 melakukan penyerangan terhadap markas musuh.

4.1.2 Pengujian pada Map Area Terbuka

Hasil pengujian pada map area terbuka dengan strategi kembali ke markas, menyerang dan bertahan dapat dlilihat pada tabel 4.1.

Tabel 4.1 Hasil pengujian dengan strategi kembali ke markas, menyerang dan bertahan dibandingkan dengan strategi menyerang dan bertahan.

Percobaan	NPC dengan strategi kembali ke markas, menyerang, dan melindungi markas	NPC dengan strategi menyerang dan melindungi markas
1	Menang	Kalah
2	Menang	Kalah
3	Menang	Kalah
4	Kalah	Menang
5	Menang	Kalah
6	Menang	Kalah
7	Kalah	Menang
8	Menang	Kalah
9	Menang	Kalah
10	Kalah	Menang

Dari hasil data tabel 4.1 didapatkan hasil bahwa dengan npc yang memiliki strategi menyerang, bertahan dan melindungi markas memiliki persentase kemenangan sebesar 70% bila dibandingkan dengan npc yang memiliki strategi menyerang dan bertahan.

4.1.3 Pengujian pada Map Area Labirin

Hasil pengujian pada map area labirin dengan strategi kembali ke markas, menyerang dan bertahan dapat dlilihat pada tabel 4.2.

Tabel 4.2 Hasil pengujian dengan strategi kembali ke markas, menyerang dan bertahan dibandingkan dengan strategi menyerang dan bertahan.

Percobaan	NPC dengan strategi kembali ke markas, menyerang, dan melindungi markas	NPC dengan strategi menyerang dan melindungi markas
1	Menang	Kalah
2	Kalah	Menang

Percobaan	NPC dengan strategi kembali ke markas, menyerang, dan melindungi markas	NPC dengan strategi menyerang dan melindungi markas
3	Menang	Kalah
4	Kalah	Menang
5	Menang	Kalah
6	Menang	Kalah
7	Kalah	Menang
8	Kalah	Menang
9	Menang	Kalah
10	Kalah	Menang

Dari hasil data tabel 4.2 didapatkan hasil bahwa dengan npc yang memiliki strategi menyerang, bertahan dan melindungi markas memiliki persentase kemenangan sebesar 50% bila dibandingkan dengan npc yang memiliki strategi menyerang dan bertahan.

4.2 Pengujian Strategi Penyerangan dengan Mengubah Parameter Kesehatan dan Battle Poin

Pengujian ini bertujuan untuk memperoleh parameter kesehatan atau *health poin*t yang tepat untuk dijadikan acuan pada saat NPC musuh mundur ke markas untuk melakukan penyembuhan. Pengujian ini dilakukan dengan membandingkan persentase kemenangan npc musuh terhadap npc pemain.

4.2.1 Pengujian dengan Menggunakan Persentase Sisa Kesehatan Sebesar 50%, 40% dan 30%

Hasil pengujian persentase kesehatan sebesar 50%, 40% dan 30% yang digunakan sebagai parameter mundur ke markas dapat dlilihat pada tabel 4.3.

Tabel 4.3 Hasil pengujian parameter sisa kesehatan

No	Kesehatan 50%		Kesehatan 40%		Kesehatan 30%	
	Musuh	**Pemain**	**Musuh**	**Pemain**	**Musuh**	**Pemain**
1	Menang	Kalah	Menang	Kalah	Kalah	Menang
2	Menang	Kalah	Menang	Kalah	Kalah	Menang
3	Menang	Kalah	Kalah	Menang	Menang	Kalah
4	Kalah	Menang	Kalah	Menang	Kalah	Menang

No	Kesehatan 50%		Kesehatan 40%		Kesehatan 30%	
	Musuh	**Pemain**	**Musuh**	**Pemain**	**Musuh**	**Pemain**
5	Menang	Kalah	Menang	Kalah	Menang	Kalah
6	Menang	Kalah	Kalah	Menang	Menang	Kalah
7	Kalah	Menang	Kalah	Menang	Kalah	Menang
8	Menang	Kalah	Menang	Kalah	Menang	Kalah
9	Menang	Kalah	Kalah	Manang	Kalah	Menang
10	Kalah	Menang	Menang	Kalah	Kalah	Menang

Dari hasil data tabel 4.3 didapatkan hasil bahwa dengan menggunakan paramater kesehatan dengan tingkat kesehatan 50% memperoleh persentase kemenangan sebersar 70%, tingkat kesehatan 40% memperoleh kemenangan sebersar 50%, dan tingkat kesehatan 30% memperoleh kemenangan sebesar 40%.

4.2.2 Pengujian dengan Membandingkan Persentase Kemenangan dengan Menggunakan Battle Poin dan Tanpa Battle Poin.

Hasil pengujian perbandingan menggunakan battle poin dan tanpa battle poin dapat dlilihat pada tabel 4.4.

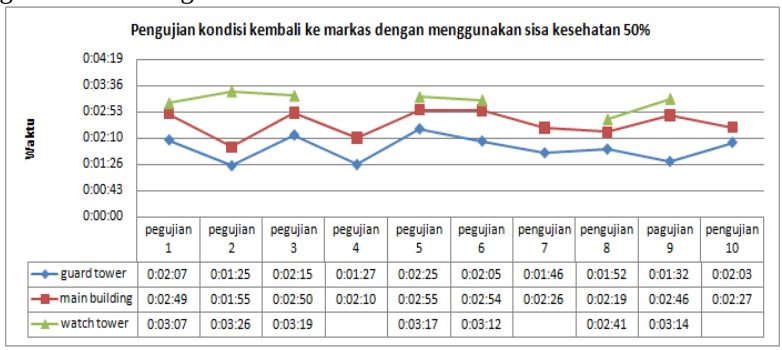
Tabel 4.4 Hasil pengujian parameter *battle* poin

No	Dengan *Battle* Poin		Tanpa *Battle* Poin		Musuh Dengan *Battle* Poin	Pemain Tanpa *Battle* Poin
	Musuh	**Pemain**	**Musuh**	**Pemain**		
1	Menang	Kalah	Kalah	Menang	Menang	Kalah
2	Kalah	Menang	Kalah	Menang	Menang	Kalah
3	Kalah	Menang	Menang	Kalah	Kalah	Menang
4	Menang	Kalah	Kalah	Menang	Menang	Kalah
5	Menang	Kalah	Menang	Kalah	Kalah	Menang
6	Kalah	Menang	Kalah	Menang	Menang	Kalah
7	Menang	Kalah	Menang	Kalah	Kalah	Menang
8	Kalah	Menang	Kalah	Menang	Kalah	Menang
9	Menang	Kalah	Kalah	Menang	Menang	Kalah
10	Kalah	Menang	Menang	Kalah	Menang	Kalah

Dari data tabel 4.4 didapatkan hasil percobaan bahwa ketika npc musuh dan npc pemain sama-sama menggunakan *battle* poin maka npc musuh memperoleh tingkat kemenangan sebesar 50% sedangkan npc pemain memperoleh kemenangan sebesar 50%, ketika npc musuh dan pemain sama-sama tidak menggunakan *battle* poin maka npc musuh memperoleh tingkat kemenangan sebesar 40% sedangkan npc pemain memperoleh kemenangan sebesar 60%, dan ketika npc musuh menggunakan *battle* poin dan npc pemain tidak menggunakan *battle* poin maka tingkat kemengan npc musuh sebesar 60% sedangkan npc pemain memperoleh kemenangan sebesar 40%.

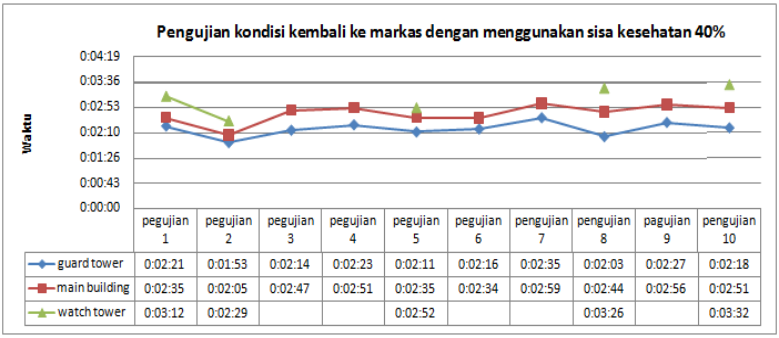
4.2.3 Pengujian dengan Menggunakan Persentase Sisa Kesehatan Sebesar 50%, 40% dan 30% dalam Bentuk Grafik

Pada pengujian ini dilakukan percobaan dengan mengubah nilai parameter *health* poin pada NPC musuh untuk mengambil keputusan kapan NPC musuh tersebut memutuskan kembali ke markasnya untuk melakukan pemulihan kesehatan. Dimana persentase yang digunakan pada percobaan ini adalah ketika NPC dengan kondisi *health* poin sebesar 50%, 40% dan 30%. Dari ketiga persentase tersebut akan diambil hasil yang paling baik, yaitu persentase kemenangan NPC musuh yang paling besar. Kemudian persentase *health* poin tersebut akan digunakan pada *rule* strategi penyerangan yang telah dibuat. Berikut ini adalah hasil pengujian persentase kesehatan sebesar 50%, 40% dan 30% yang digunakan sebagai parameter mundur ke markas dapat dilihat pada gambar 4.5, 4.6 dan 4.7 yang ditunjukkan dengan grafik waktu bangunan hancur.



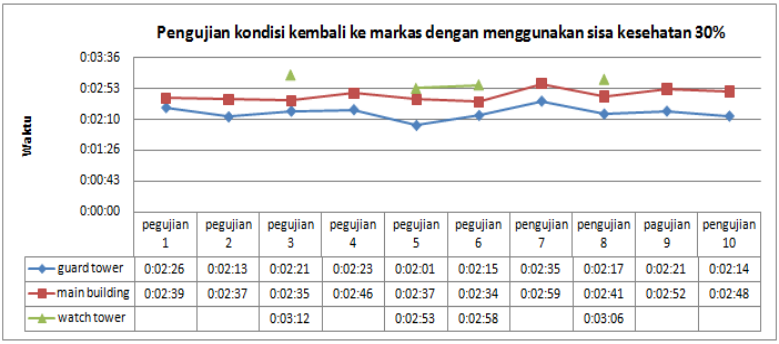
	pegujian 1	pegujian 2	pegujian 3	pegujian 4	pegujian 5	pegujian 6	pengujian 7	pengujian 8	pagujian 9	pengujian 10
guard tower	0:02:07	0:01:25	0:02:15	0:01:27	0:02:25	0:02:05	0:01:46	0:01:52	0:01:32	0:02:03
main building	0:02:49	0:01:55	0:02:50	0:02:10	0:02:55	0:02:54	0:02:26	0:02:19	0:02:46	0:02:27
watch tower	0:03:07	0:03:26	0:03:19		0:03:17	0:03:12		0:02:41	0:03:14	

Gambar 4.5 Grafik waktu bangunan hancur dengan menggunakan sisa kesehatan 50%



	pegujian 1	pegujian 2	pegujian 3	pegujian 4	pegujian 5	pegujian 6	pengujian 7	pengujian 8	pagujian 9	pengujian 10
guard tower	0:02:21	0:01:53	0:02:14	0:02:23	0:02:11	0:02:16	0:02:35	0:02:03	0:02:27	0:02:18
main building	0:02:35	0:02:05	0:02:47	0:02:51	0:02:35	0:02:34	0:02:59	0:02:44	0:02:56	0:02:51
watch tower	0:03:12	0:02:29			0:02:52			0:03:26		0:03:32

Gambar 4.6 Grafik waktu bangunan hancur dengan menggunakan sisa kesehatan 40%



	pegujian 1	pegujian 2	pegujian 3	pegujian 4	pegujian 5	pegujian 6	pengujian 7	pengujian 8	pagujian 9	pengujian 10
guard tower	0:02:26	0:02:13	0:02:21	0:02:23	0:02:01	0:02:15	0:02:35	0:02:17	0:02:21	0:02:14
main building	0:02:39	0:02:37	0:02:35	0:02:46	0:02:37	0:02:34	0:02:59	0:02:41	0:02:52	0:02:48
watch tower			0:03:12		0:02:53	0:02:58		0:03:06		

Gambar 4.7 Grafik waktu bangunan hancur dengan menggunakan sisa kesehatan 30%

Dari gambar 4.5, 4.6 dan 4.7 didapatkan hasil dengan menggunakan sisa kesehatan 50% untuk parameter kemabali ke markas didapatkan persentase kemenangan yang paling besar.

4.3 Pengujian Rule yang Lebih Mengacu pada Battle poin dan Rule yang Lebih Mengacu pada Health Poin

Pengujian ini bertujuan untuk membandingkan *rule* yang lebih mengacu pada *battle* poin dan *rule* yang lebih mengacu pada *health* poin. Rule yang lebih mengacu pada battle poin dapat dilihat pada tabel 4.5 dan rule yang lebih mengacu pada health poin dapat dilihat pada tabel 4.6.

Tabel 4.5 Daftar *rule* yang lebih mengacu pada *battle* poin

KONDISI		**KEPUTUSAN**
Battle Poin	Health Poin	Menyerang Musuh
BP_E < 50%	HP_E < 50%	Ya
BP_E < 50%	HP_E >= 50%	Ya
BP_E >= 50%	HP_E < 50%	Ya
BP_E >= 50%	HP_E >= 50%	Ya
BP_E < 50%	HP_P < 50%	Ya
BP_E < 50%	HP_P >= 50%	Tidak
BP_E >= 50%	HP_P < 50%	Ya
BP_E >= 50%	HP_P >= 50%	Ya
BP_P < 50%	HP_E < 50%	Tidak
BP_P < 50%	HP_E >= 50%	Ya
BP_P >= 50%	HP_E < 50%	Tidak
BP_P >= 50%	HP_E >= 50%	Tidak
BP_P < 50%	HP_P < 50%	Tidak
BP_P < 50%	HP_P >= 50%	Tidak
BP_P >= 50%	HP_P < 50%	Tidak
BP_P >= 50%	HP_P >= 50%	Tidak

Tabel 4.6 Daftar *rule* yang lebih mengacu pada *health* poin

KONDISI		**KEPUTUSAN**
Battle Poin	Health Poin	Menyerang Musuh
BP_E < 50%	HP_E < 50%	Ya
BP_E < 50%	HP_E >= 50%	Ya
BP_E >= 50%	HP_E < 50%	Ya
BP_E >= 50%	HP_E >= 50%	Ya
BP_E < 50%	HP_P < 50%	Tidak
BP_E < 50%	HP_P >= 50%	Tidak
BP_E >= 50%	HP_P < 50%	Ya
BP_E >= 50%	HP_P >= 50%	Tidak
BP_P < 50%	HP_E < 50%	Ya
BP_P < 50%	HP_E >= 50%	Ya
BP_P >= 50%	HP_E < 50%	Tidak

KONDISI		KEPUTUSAN
Battle Poin	Health Poin	Menyerang Musuh
BP_P >= 50%	HP_E >= 50%	Ya
BP_P < 50%	HP_P < 50%	Tidak
BP_P < 50%	HP_P >= 50%	Tidak
BP_P >= 50%	HP_P < 50%	Tidak
BP_P >= 50%	HP_P >= 50%	Tidak

Berikut ini adalah hasil percobaan sebanyak 10 kali dari *rule* yang lebih mengacu pada *battle* poin dan *rule* yang lebih mengacu pada *health* poin dengan melawan NPC pemain yang sama dapat dilihat pada tabel 4.7.

Tabel 4.7 Hasil percobaan *rule battle* poin dan *rule health* poin.

Percobaan	***Rule Battle* Poin**	***Rule Health* Poin**
1	Menang	Menang
2	Menang	Kalah
3	Menang	Kalah
4	Kalah	Menang
5	Menang	Menang
6	Menang	Kalah
7	Kalah	Menang
8	Menang	Menang
9	Kalah	Kalah
10	Menang	Menang

Dari hasil data tabel 4.7 didapatkan hasil bahwa dengan menerapkan *rule* yang lebih mengacu pada *battle* poin memiliki persentase tingkat kemenangan sebesar 70% sedangkan dengan menerapkan *rule* yang lebih mengacu pada *health* poin memiliki persentase tingkat kemenangan sebesar 60%.

4.4 Ketercapaian Konsep

Pada bagian ini akan dibandingkan antara strategi penyerangan yang dibuat dengan realisasi penerapannya pada permainan *Fantasy*

Chronicles, selain itu akan disampaikan pula kondisi ketercapaian. Ketercapaian strategi penyerangan ini dapat dilihat pada tabel 4.8.

Tabel 4.8 Ketercapaian konsep

No.	Konsep	Terlaksana	Keterangan
1	NPC Menyerang	Ya	NPC dapat menyerang sesuai dengan skenario.
2	NPC Bertahan	Ya	NPC dapat bertahan sesuai dengan skenario.
3	NPC Kembali ke Markas	Ya	NPC dapat bersiaga sesuai dengan skenario.
4	NPC Bertarung	Ya	NPC dapat bertarung dengan musuh jika *battle* poin dan *health* poin mencukupi.
5	NPC Kabur	Ya	NPC dapat kabur jika *battle* poin dan *health* poin tidak mencukupi.

Dari tabel 4.8. jika dirata rata maka ketercapaian konsep strategi penyerangan pada permainan ini adalah sekitar 100%.

Halaman ini sengaja dikosongkan

BAB 5
PENUTUP

Setelah dilakukan perancangan, implementasi, percobaan dan pengujian, akhirnya diperoleh beberapa kesimpulan serta kritik dan saran untuk pengembangan selanjutnya.

5.1 Kesimpulan

Berdasarkan hasil pengujian sistem dalam tugas akhir ini dapat diambil beberapa kesimpulan sebagai berikut:

1. NPC musuh dengan sistem battle poin memiliki tingkat kemenangan sebesar 60% dibandingkan NPC pemain tanpa menggunakan sistem battle poin yaitu sebesar 40%.
2. NPC dengan strategi menyerang, bertahan dan kembali ke markas memiliki tingkat kemenangan sebesar 70% bila dibandingkan dengan NPC dengan strategi menyerang dan bertahan yaitu sebesar 30%, semakin dekat markas dengan area pertempuran maka semakin besar tingkat kemenangannya.
3. Strategi kembali ke markas untuk melakukan pemulihan kesehatan ketika kondisi *health* poin sisa 50% memiliki tingkat kemenangan sebesar 70% bila dibandingkan dengan kondisi *health* poin sisa 40% hanya sebesar 50% dan kondisi *health* poin 30% hanya sebesar 40%.

5.2 Saran

Untuk pengembangan lebih lanjut mengenai tugas akhir ini, disarankan untuk melakukan beberapa langkah lanjutan:

1. Menambahkan strategi penyerangan yang lebih bervariasi.
2. Menambahkan jenis-jenis karakter dengan tipe yang berbeda agar strategi penyerangan lebih bervariasi.
3. Menambahkan skenario agar penyerangan lebih bervariasi.

Halaman ini sengaja dikosongkan

DAFTAR PUSTAKA

[1] Yunifa Miftahul Arif, Mochamad Hariadi, Supeno Mardi S.N. "*Strategi Menyerang pada Game FPS Menggunakan Hierarchy Finite Machine dan Logika Fuzzy"*.

[2] Ibrohim Yofid Fananda, Mochamad Hariadi, Supeno Mardi. 2011. "*Simulasi Multi-Agent Dengan Hexagonal Grid Menggunakan Metode Knowledge-Based System*".

[3] Marcus Svensson, Johan Hagelback. 2015. "*Dynamic Strategy in Real-Time Strategy Games"*.

[4] Anderson R.Tavares, Hector Azpurua, Luiz Chaimowicz. 2014. *"Evolving swarm intelligence for task allocation in a real time strategy game"*.

[5] Febrian Bahari Adi, Mochammad Hariadi, I Ketut Eddy Purnama. 2013. *"Simulasi Perilaku Tempur Pada Sekumpulan NPC Berbasis Boid*".

[6] Aleksandar Micic, Davio Arnarsson, Vignir Jonsson. 2011. "*Developing Game AI For Real-Time Strategy Game Starcraft*".

Halaman ini sengaja dikosongkan

LAMPIRAN

```
using UnityEngine;
using System.Collections;
using System.Collections.Generic;

public class AIPathMusuh6 : MonoBehaviour
{
	NavMeshAgent agent;
	public GameObject target, target1, target2,
target3;
	public GameObject rumahMundur, rumah1, rumah2,
rumah3;
	public bool adaMusuh1=false, adaMusuh2=false,
adaMusuh3=false;
	public float
jarakSerang=5f,jarakAttack=2f,jarakAttackBangunan=3.5f
;
	public enum state { IDLE, ATTACKING, DEFENDING,
FLEE, BATTLE, BACK }
	public state State;
	private bool isBattle=false, isAttacking=false,
isDefending=false, isFlee=false, isBack=false,
reachGoal=true;
	public bool sudahMaju=false, recovery=false;
	private string namaRumah = "";
	private Vector3 goal;
	private float
delayAttack=1f,nextAttack=0f,SpeedDampTime=0.25f,
jarak,jarak1,jarak2,jarak3,jarak4,
jRmhMundur,jRmh1,jRmh2,jRmh3;
	private totalBattlePoint bp;
	private EnemyUnitManager enemyManager;
	protected Animator m_Animator;
	private float curHP, maxHP, percentOfHP;
	private Health hp;
	private float selBP=0,selHP=0;
	private int hslBP=0,hslHP=0;
	private bool ms = false;

	void Start()
	{
```

```
            enemyManager =
GameObject.FindGameObjectWithTag("PlayerUnitManager").
GetComponent<EnemyUnitManager>();
            bp = GetComponent<totalBattlePoint>();
            m_Animator =
this.gameObject.GetComponent<Animator>();
            agent = GetComponent<NavMeshAgent>();
            enemyManager.AddEnemyUnit(gameObject);
            hp = GetComponentInParent<Health>();
            maxHP = hp.curHp; curHP = hp.curHp;
            namaRumah = daftarRumah();
      }

      void Update()
      {
            curHP = hp.curHp;
            percentOfHP = curHP/maxHP;
            if(percentOfHP < 0.5f)
m_Animator.SetFloat("Sakit",0.65f);
            target=findClosestVisibilityUnit();
target1=findClosestVisibilityBuilding();
            target2=findClosestVisibilityFriend();
target3=findClosestVisibilityFriendBuilding();
            rumahMundur=mundurFunc();
            if(target!=null) jarak =
Vector3.Distance(transform.position,
target.transform.position);
            if(target1!=null) jarak1 =
Vector3.Distance(transform.position,
target1.transform.position);
            if(target2!=null) jarak2 =
Vector3.Distance(transform.position,
target2.transform.position);
            if(target3!=null) jarak3 =
Vector3.Distance(transform.position,
target3.transform.position);
            if(!reachGoal) jarak4  =
Vector3.Distance(transform.position, goal);
            if(rumah1!=null) jRmh1 =
Vector3.Distance(transform.position,
rumah1.transform.position);
            if(rumah2!=null) jRmh2 =
Vector3.Distance(transform.position,
rumah2.transform.position);
```

```
            if(rumah3!=null) jRmh3 =
Vector3.Distance(transform.position,
rumah3.transform.position);
            if(rumahMundur!=null) jRmhMundur =
Vector3.Distance(transform.position,
rumahMundur.transform.position);

            if(target!=null &&
bp.battlePointsEnemy!=0 && bp.battlePointsEnemy >=
bp.battlePointsPlayer)
            {
                  selBP = (bp.battlePointsEnemy-
bp.battlePointsPlayer) / bp.battlePointsEnemy;
                  if(selBP < 0.5f) hslBP = 0; else
hslBP = 1;
            }
            else if(target!=null &&
bp.battlePointsPlayer!=0 && bp.battlePointsPlayer >=
bp.battlePointsEnemy)
            {
                  selBP = (bp.battlePointsPlayer-
bp.battlePointsEnemy) / bp.battlePointsPlayer;
                  if(selBP < 0.5f) hslBP = 2; else
hslBP = 3;
            }
            if(target!=null && bp.hpEnemy!=0 &&
bp.hpEnemy >= bp.hpPlayer)
            {
                  selHP = (bp.hpEnemy-bp.hpPlayer) /
bp.hpEnemy;
                  if(selHP < 0.5f) hslHP = 0; else
hslHP = 1;
            }
            else if(target!=null && bp.hpPlayer!=0
&& bp.hpPlayer >= bp.hpEnemy)
            {
                  selHP = (bp.hpPlayer-bp.hpEnemy) /
bp.hpPlayer;
                  if(selHP < 0.5f) hslHP = 2; else
hslHP = 3;
            }

            switch(hslBP)
            {
```

```
                case 0:
                       if(hslHP==0) ms=true;
                       else if(hslHP==1) ms=true;
                       else if(hslHP==2) ms=true;
                       else if(hslHP==3) ms=false;
                       break;
                case 1:
                       if(hslHP==0) ms=true;
                       else if(hslHP==1) ms=true;
                       else if(hslHP==2) ms=true;
                       else if(hslHP==3) ms=true;
                       break;
                case 2:
                       if(hslHP==0) ms=false;
                       else if(hslHP==1) ms=true;
                       else if(hslHP==2) ms=false;
                       else if(hslHP==3) ms=false;
                       break;
                case 3:
                       if(hslHP==0) ms=true;
                       else if(hslHP==1) ms=true;
                       else if(hslHP==2) ms=true;
                       else if(hslHP==3) ms=true;
                       break;
                }

                if(target!=null)
                {
                       if(percentOfHP < 0.5f){State =
state.BACK; recovery=true;}
                       else
                       {
                              if(jarak<=jarakSerang)
                              {
                                     if(target!=null &&
jarak<=jarakAttack){ State=state.BATTLE;
status(true,false); }
                                     else
                                     {
                                            if(ms){
State=state.BATTLE; status(true,false); }
                                            else{
State=state.FLEE; status(false,true); }
                                     }
```

```
                                    isDefending = false;
                              }
                              else
                              {
                                    if(isDefending &&
jarak4<jarak1){State = state.DEFENDING;}
                                    else if(!recovery){
status(false,false); State = state.ATTACKING; }
                              }
                        }
                }
                else if(target1!=null)
                {status(false,false); State =
state.ATTACKING;}

                switch(State)
                {
                case state.ATTACKING: if(!isFlee &&
!isBattle) { AttackingState(); } break;
                case state.DEFENDING: if(!isFlee &&
!isBattle) { DefendingState(); } break;
                case state.BATTLE: BattleState(); break;
                case state.FLEE: FleeState(); break;
                case state.BACK: BackState(); break;
                case state.IDLE: IdleState(); break;
                default: IdleState(); break;
                }
        }

        public void isiHP()
        {
                float hpPlus2 = 20;
                if(rumahMundur.name == namaRumah)
hpPlus2 = 20;
                else hpPlus2 = 10;
                if(jRmhMundur <= 7f && jarak >
jarakSerang)
                {
                        if(hp.curHp < maxHP)
                        {
                        hp.curHp = hp.curHp + hpPlus2;
                        if(hp.curHp > maxHP)
                        {recovery=false; hp.curHp =
maxHP;}
```

```
                }
            }
        }
        public void BackState()
        {
            if(jarak3>4f)
            {

        agent.SetDestination(rumahMundur.transform.posi
tion);

        m_Animator.SetFloat("Speed",0.15f,SpeedDampTime
, Time.deltaTime);
            }
            else State = state.IDLE;
        }
        public void AttackOrder()
        {
            isAttacking=true;
            if(!isDefending){ State =
state.ATTACKING; }
        }
        public void DefendOrder(Vector3 newGoal)
        { goal=newGoal; isDefending=true;
reachGoal=false; }
        public void Stay()
        {  isDefending=false; State = state.ATTACKING;
AttackingState(); }

        private void status(bool battle, bool flee)
        { isBattle=battle; isFlee=flee; }

        private void serang(GameObject musuh)
        {

        transform.LookAt(musuh.transform.position);
//agent.Stop();

        m_Animator.SetFloat("Speed",0.0f,SpeedDampTime,
Time.deltaTime);
            if(Time.time > nextAttack)
            { nextAttack = Time.time + delayAttack;
m_Animator.SetTrigger("Serang"); }
        }
```

```
	private void IdleState()
	{
		status(false,false); isDefending=false;
reachGoal=true; agent.Stop();

	m_Animator.SetFloat("Speed",0.0f,SpeedDampTime,
Time.deltaTime);
		if(target!=null && jarak<=jarakAttack){
State=state.BATTLE; status(true,false); }
		else if(percentOfHP >= 0.5f &&
!recovery){status(false,false);
State=state.ATTACKING;}
		else if(percentOfHP ==
1){recovery=false; status(false,false);
State=state.ATTACKING;}
	}

	private void AttackingState()
	{
		if(target1!=null)
agent.SetDestination(target1.transform.position);
		if(target1!=null && jarak1>jarakSerang){
m_Animator.SetFloat("Speed",0.15f,SpeedDampTime,
Time.deltaTime); }
		else if(target1!=null){ serang(target1);
if(this.gameObject.name=="MAGE(Clone)") agent.Stop();
}
		else{ target1=null; State=state.IDLE; }
	}

	private void DefendingState()
	{
		agent.SetDestination(goal);
		if(!reachGoal)
		{

	if(Vector3.Distance(this.transform.position,
goal) > 3)
			{
m_Animator.SetFloat("Speed",0.15f,SpeedDampTime,
Time.deltaTime); }
```

```
                    else{
m_Animator.SetFloat("Speed",0.0f,SpeedDampTime,
Time.deltaTime); reachGoal=true; agent.Stop(); }
             }
             else{ status(true,false);
State=state.BATTLE; }
       }

       private void FleeState()
       {
             if(target!=null && jarak<=jarakSerang &&
jarak>=jarakAttack)
             {
                    if(target2!=null)
                    {

       agent.SetDestination(target2.transform.position
);
                           if(target2!=null &&
jarak2>jarakAttackBangunan){
m_Animator.SetFloat("Speed",0.15f,SpeedDampTime,
Time.deltaTime); }
                           else{
m_Animator.SetFloat("Speed",0.0f,SpeedDampTime,
Time.deltaTime); agent.Stop(); }
                    }
                    else
                    {

       transform.LookAt(target.transform.position);
                           transform.Rotate(0,180,0);
transform.Translate(Vector3.forward * 1 *
Time.deltaTime);

       m_Animator.SetFloat("Speed",0.15f,SpeedDampTime
, Time.deltaTime);
                    }
             }
             else if(target!=null &&
jarak<jarakAttack) { status(true,false); State =
state.BATTLE; }
             else { status(false,false); State =
state.IDLE; }
       }
```

```
        private void BattleState()
        {
                if(target!=null && jarak<=jarakSerang)
                {
                        if(ms)
                        {
        agent.SetDestination(target.transform.position)
; status(true,false);
                                if (target!=null && jarak >
jarakSerang) {
m_Animator.SetFloat("Speed",0.15f,SpeedDampTime,
Time.deltaTime); }
                                else if(target!=null) {
serang(target);
if(this.gameObject.name=="MAGE(Clone)") agent.Stop();
}
                                else { target=null;
isBattle=false; State=state.IDLE; }
                        }
                        else if(target!=null && isBattle
&& jarak<=jarakAttack) { serang(target);
if(this.gameObject.name=="MAGE(Clone)") agent.Stop();
}
                        else if(target==null) {
target=null; isBattle=false; State=state.IDLE; }
                        else { status(false,true); State =
state.FLEE; }
                }
                else { target=null; status(false,false);
State=state.IDLE;}
        }

        GameObject findClosestVisibilityUnit()
        {
                GameObject[]objectArray;
                objectArray =
GameObject.FindGameObjectsWithTag("PlayerUnits");
                float distance = Mathf.Infinity;
                Vector3 position = transform.position;
                foreach(GameObject currentObject in
objectArray)
                {
```

```
                    Vector3 distanceCheck =
currentObject.transform.position - position;
                    float currentDistance =
distanceCheck.sqrMagnitude;
                    if (currentDistance < distance)
                    { target = currentObject; distance
= currentDistance; }
             } return target;
       }
       GameObject findClosestVisibilityBuilding()
       {
             GameObject[]objectArray1;
             objectArray1 =
GameObject.FindGameObjectsWithTag("PlayerBuilding");
             float distance1 = Mathf.Infinity;
             Vector3 position1 = transform.position;
             foreach(GameObject currentObject1 in
objectArray1)
             {
                    Vector3 distanceCheck1 =
currentObject1.transform.position - position1;
                    float currentDistance1 =
distanceCheck1.sqrMagnitude;
                    if (currentDistance1 < distance1)
                    { target1 = currentObject1;
distance1 = currentDistance1; }
             } return target1;
       }
       GameObject findClosestVisibilityFriend()
       {
             GameObject[]objectArray2;
             objectArray2 =
GameObject.FindGameObjectsWithTag("Enemies");
             List<GameObject> list = new
List<GameObject>(objectArray2);
             list.Sort(ByDistance);
             if(objectArray2.Length>1)
                    target2 = objectArray2[1];
             else target2=null; return target2;
       }
GameObject findClosestVisibilityFriendBuilding ()
       {
             GameObject[]objectArray3;
```

```
            objectArray3 =
GameObject.FindGameObjectsWithTag("EnemyBuilding");
            float distance3 = Mathf.Infinity;
            Vector3 position3 = transform.position;
            foreach(GameObject currentObject3 in
objectArray3)
            {
                  Vector3 distanceCheck3 =
currentObject3.transform.position - position3;
                  float currentDistance3 =
distanceCheck3.sqrMagnitude;
                  if (currentDistance3 < distance3)
                  { target3 = currentObject3;
distance3 = currentDistance3; }
            } return target3;
      }
      int ByDistance(GameObject a, GameObject b)
      {
            float dstToA =
Vector3.Distance(transform.position,
a.transform.position);
            float dstToB =
Vector3.Distance(transform.position,
b.transform.position);
            return dstToA.CompareTo(dstToB);
      }
      int HitungTotalEnemy()
      {
            GameObject[]objectArray3;
            objectArray3 =
GameObject.FindGameObjectsWithTag("Enemies");
            int jmlEnemyUnit = objectArray3.Length;
            return jmlEnemyUnit;
      }
      GameObject mundurFunc()
      {
            if(rumah1!=null && !adaMusuh1) { return
rumah1; }
            else if(rumah2!=null && !adaMusuh2)
            {
                  if(rumah3!=null && !adaMusuh3)
                  {
                  if(jRmh2<jRmh3) return rumah2;
                  else return rumah3;
```

```
                }
                else return rumah2;
            }
            else if(rumah3!=null && !adaMusuh3)
            {
                if(rumah2!=null && !adaMusuh2)
                {
                if(jRmh2<jRmh3) return rumah2;
                else return rumah3;
                }
                else return rumah3;
            }
            else { return null; }
        }
        string daftarRumah()
        {
        string rmhAsal = "";
        if(this.gameObject.name=="GOBLIN2(Clone)")
        {
        rumah1=GameObject.Find("Wartent");
        rumah2=GameObject.Find("Guard_tower");
        rumah3=GameObject.Find("Watchtower");
        rmhAsal = "Wartent";
        return rmhAsal;
        }
        else if(this.gameObject.name=="MAGE(Clone)")
        {
        rumah1=GameObject.Find("Guard_tower");
        rumah2=GameObject.Find("Wartent");
        rumah3=GameObject.Find("Watchtower");
        rmhAsal = "Guard_tower";
        return rmhAsal;
        }
        else if(this.gameObject.name=="Troll(Clone)")
        {
        rumah1=GameObject.Find("Watchtower");
        rumah2=GameObject.Find("Guard_tower");
        rumah3=GameObject.Find("Wartent");
        rmhAsal = "Watchtower";
            return rmhAsal;
            }
            else{ return rmhAsal; }
        }
}
```

BIOGRAFI PENULIS

I Putu Widya Wiranata. lahir di Surabaya pada tanggal 8 Mei 1991. Menyelesaikan pendidikan dasar di SDN Perak Barat 1 Surabaya. Kemudian menempuh jalur pendidikan di SLTP Negeri 2 Surabaya dan SMA Negeri 21 Surabaya tahun 2006 hingga tahun 2009. Melanjutkan pendidikan ke jenjang perkuliahan di Jurusan D3 Teknik Informatika PENS-ITS Surabaya dari tahun 2009 sampai 2012. Dan melanjutkan kuliah ke jenjang sarjana di Lintas Jalur ITS jurusan Teknik Elektro prodi Teknik Komputer dan Telematika.

Halaman ini sengaja dikosongkan